Integrity Verification Mechanism of Sensor Data Based on Bilinear Map Accumulator

YONGJUN REN, JIAN QI, and YEPENG LIU, School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, China
JIN WANG, School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha, China
GWANG-JUN KIM, Chonnam National University, Dept. of Computer Engineering, 96-1 Mt, Dundeok-dong, Yeosu-si, Jeollanam-do, Gwangju, Korea

With the explosive growth in the number of IoT devices, ensuring the integrity of the massive data generated by these devices has become an important issue. Due to the limitation of hardware, most past data integrity verification schemes randomly select partial data blocks and then perform integrity validation on those blocks instead of examining the entire dataset. This will result in that unsampled data blocks cannot be detected even if they are tampered with. To solve this problem, we propose a new and effective integrity auditing mechanism of sensor data based on a bilinear map accumulator. Using the proposed approach will examine all the data blocks in the dataset, not just some of the data blocks, thus, eliminating the possibility of any cloud manipulation. Compared with other schemes, our proposed solution has been proved to be highly secure for all necessary security requirements, including tag forgery, data deletion, replacement, replay, and data leakage attacks. The solution reduces the computational and storage costs of cloud storage providers and verifiers, and also supports dynamic operations for data owners to insert, delete, and update data by using a tag index table (TIT). Compared with existing schemes based on RSA accumulator, our scheme has the advantages of fast verification and witness generation and no need to map data blocks to prime numbers. The new solution supports all the characteristics of a data integrity verification scheme.

CCS Concepts: • **Security and privacy → Cryptography**; **Public key (asymmetric) techniques**; **Public key encryption**; • **Applied computing → Enterprise computing**; **Enterprise data management**; • **Theory of computation → Design and analysis of algorithms**; **Data structures design and analysis**; **Pattern matching**;

Additional Key Words and Phrases: Sensor data storage, accumulator, data integrity verification

ACM Reference format:
Yongjun Ren, Jian Qi, Yepeng Liu, Jin Wang, and Gwang-Jun Kim. 2021. Integrity Verification Mechanism of Sensor Data Based on Bilinear Map Accumulator. *ACM Trans. Internet Technol.* 21, 1, Article 5 (February 2021), 19 pages.
https://doi.org/10.1145/3380749

This research was funded by NSFC (61772280, 61772454, 61811530332, 61811540410), and the PAPD fund from NUIST.
Author's addresses: Y. Ren, J. Qi, Y. Liu, School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, China; emails: renyj100@126.com, qijian19940420@163.com, wind.bell@live.cn; J. Wang (corresponding author), School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha, China; email: jinwang@csust.edu.cn; G.-J. Kim, Chonnam National University, Dept. of Computer Engineering, 96-1 Mt, Dundeok-dong, Yeosu-si, Jeollanam-do, Gwangju, Korea; email: kgj@chonnam.ac.kr.

1533-5399/2021/02-ART5 $15.00
https://doi.org/10.1145/3380749

1 INTRODUCTION

The latest generation of cellular mobile communication technology "5G" heralds a major change in the network space. Although it was not born for the Internet of Things (IoT), it is still considered by the industry to be the main driver of the growth of the IoT [1, 2]. With the development of Industry 4.0 [3], Internet of Vehicles, intelligent buildings, and smart cities [4, 5], we use the IoT to connect the physical world with the digital world and conduct valuable analysis of massive amounts of data [6, 7]. 5G can provide large connectivity, high reliability, and real time, as well as low power or deep coverage, which are all needed for the IoT. Therefore, it is foreseeable that with the launch of 5G technology, the number of IoT devices will develop in a blowout manner [8]. On the other hand, the development of these devices is transforming people's lives while generating massive amounts of data all the time. To analyze this data in real time, it is necessary to reverse the analysis workflow. For example, a new service station built between the local and the cloud is a key element of the distributed data center architecture. In this model, endpoints are located at the edge of the network [9], collecting data from local IoT devices and sensors [10]. They then process or analyze data on site, saving and pushing sensor data to the data center, as shown in Figure 1.

In this model, we have done a lot of research on cloud storage [11], edge computing [12, 13], and wireless sensor networks [14, 15]. The mobile Internet has accelerated the big data process, and big data has promoted the development of cloud computing. The history of technology development over the past decade, mobile Internet, big data and cloud computing have written a great deal [16]. However, ensuring the security and integrity of massive data is another issue that cannot be delayed [17–18].

In the entire edge computing network, the relay device and the cloud server process and store large amounts of data from different devices [19, 20], providing the data owner with the ability to remotely manage data while saving the storage space of the IoT device. However, in increasingly complex links, data may be intentionally or unintentionally corrupted. Data owners are also concerned that uploaded data could be altered or deleted without their knowledge or permission [21]. To resolve this issue, the verifier must periodically perform an integrity check on the data to verify ownership of the data and detect any unauthorized modifications [22].

Most currently available data integrity check schemes randomly select partial data blocks and then perform integrity validation on those blocks instead of examining the entire dataset. However, for an explosive number of IoT devices, probabilistic solutions are not enough. First, as its name suggests, probabilistic data integrity verification does not ensure that the data is 100% unmodified. Secondly, in the face of massive data, the computing resources and storage resources of edge nodes [23] and even cloud servers are very weak [24, 25]. A verifier using the deterministic approach will examine all the data blocks in the dataset instead of just checking a few, thus eliminating the possibility of any cloud manipulation [26]. However, in practice, this means when cloud storage providers validate large datasets, the computational overhead required will be very high. Another limitation of current schemes is that the number of data integrity verifications for data owners is limited.

Therefore, considering the shortcomings of the existing approaches, our scheme proposes a new model that uses bilinear-map accumulators to overcome these two limitations. This is a deterministic approach, and the verifier will examine all the data blocks in the dataset, eliminating any possibility of cloud manipulation. Using our scheme, a verifier will be able to perform an unlimited number of data integrity verifications on the entire dataset while maintaining computational overhead at an acceptable level. In addition, the new solution supports features such as block-less verification, public auditability of data, and dynamic data manipulation. The solution supports dynamic operations by using a tag index table (TIT) based on provable multi-copy dynamic data in a

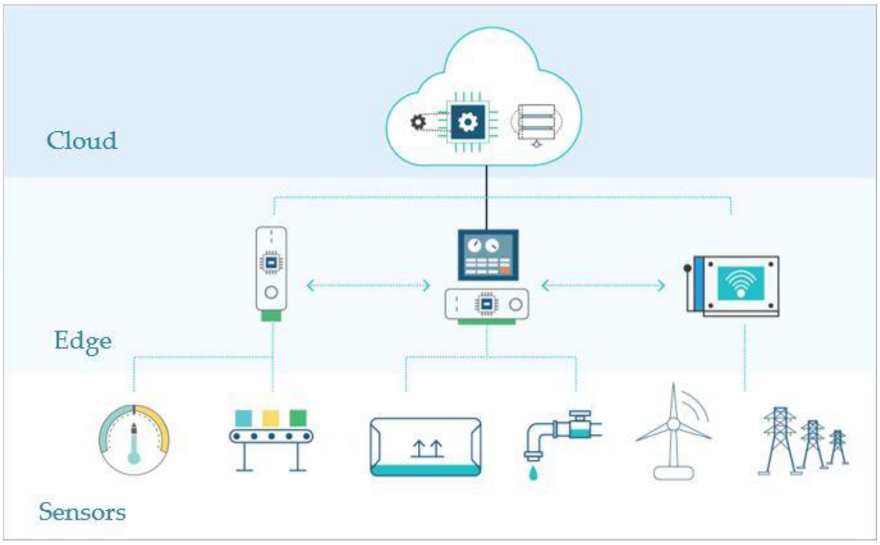

Fig. 1. Edge computing architecture.

cloud computing system. Compared with other schemes, our proposed solution has been proved to be highly secure for all necessary security requirements, including data deletion, tag forgery, data replacement, data leakage, and data replay attacks. The solution reduces the computational and storage costs of cloud storage providers and verifiers. Therefore, our proposed solution can meet the data security and integrity requirements [27] in the IoT environment [28].

Our contribution. In this article, we propose a new and effective sensor data storage integrity verification scheme based on bilinear pair accumulator. Our scheme meets all cloud data integrity characteristics, such as dynamic data handing, unrestricted challenge frequency, and public auditability. The solution reduces the computational and storage costs of cloud storage providers and verifiers. Finally, our proposed solution has been proved to be highly secure for all necessary security requirements, including data deletion, tag forgery, data replacement, data leakage, and data replay attacks.

The roadmap for the article is as follows: Section 2 introduces the related work we have done. Section 3 of this article shows our preliminaries knowledge of the scheme; Section 4 illustrates the construction of the scheme; Section 5 shows how the solution satisfies the characteristics of the data integrity verification scheme; Section 6 discusses that our solution is highly secure for all necessary security requirements; Section 7 analyzes the performance of our proposed scheme. Finally, the text concludes in Section 8.

2 RELATED WORK

In this section, we show the relevant work in more detail we have done so far. First, we introduced the development of data integrity deterministic verification schemes and the deficiencies in each scheme. After that, we mainly introduce the basic concept of the accumulator used in this article and the evolution of the accumulator and its main application.

2.1 Development of the Integrity Verification Scheme

Early schemes are faster because they validate data integrity based on a set of randomly selected blocks, while deterministic schemes are more reliable because they provide a 100% data possession and integrity guarantees. Due to the increasing demand for reliable data integrity check schemes, many recently proposed data integrity check schemes are applying deterministic methods. Caronni et al. [29] proposed a simple publicly verifiable data integrity verification mechanism, but does not support dynamic operation processing. In his proposed solution, data owners were unable to insert, delete, and update data stored at cloud storage providers. So in order to verify the authenticity of the data, the data owner needs to save a copy of the data locally. Deswarte et al. proposed another scheme [30] for key exchange based on the Diffie-Hellman cryptographic protocol. The scheme reports two main limitations. First, because CSP computes exponentiation on the entire data file during each verification process, the high computational overhead is incurred. This is especially true when the file is large. Moreover, the number of challenges that a verifier can perform when performing data integrity verification is limited. Filho et al. [31] described a protocol based on a homomorphic RSA-based secure hash function, which prevents 'cheating' in a data transfer transaction. His proposed solution supports the verifier to perform an unlimited number of data integrity verifications, but it has a high computational overhead and does not support dynamic data operations. Sebe et al. [32] proposed a remote data possession checking protocol based on the Diffie-Hellman idea. Their solution supports an unlimited number of verifications while reducing the computational overhead of cloud storage providers. However, this solution does not support dynamic data operations and public auditing. Barsoum et al. [33] proposed an efficient multi-copy provable data possession (EMC-PDP) protocol. This solution can effectively protect all data copies of cloud storage providers, but it is less efficient in terms of storage overhead and communication costs. Hao et al. [34] proposed a remote data integrity verification scheme that satisfies all of the proposed features. Although mentioned in the article, the author did not mention how their solution supports data dynamics. In addition, since a large number of modular exponential operations are implemented in this scheme, the verifier takes a long time in the setup phase and the proof generation phase. Furthermore, their solution does not prevent any data leakage from malicious providers. Recently, Khedr et al. [35] proposed a key data integrity verification scheme based on an RSA accumulator. Their solution also satisfies all of the above features, but since the input field of the RSA accumulator is limited to prime numbers, the scheme needs to map each data segment to a prime number.

2.2 Cryptographic Accumulator

The accumulator was originally proposed by Benaloh and de Mare [36] and is defined as a one-way hash function with quasi-exchange properties. That is, if for all $x \in X$ and all $y_1, y_2 \in Y$, the one-way hash function $h : X \times Y \rightarrow X$ satisfies the quasi-exchange property:

$$h(h(x, y_1), y_2) = h(h(x, y_2), y_1)$$

The accumulator scheme allows all elements x_j in the finite set $X = \{x_1, \ldots, x_n\}$ to be accumulated into a compact value acc_X; acc_X does not depend on the accumulation of x_j order. Select $g \in G$ as the base and the accumulator is defined as:

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

By calculating the witness wit_{x_i} of each element $x_i \in X$, verify $h(wit_{x_i}, x_i) = acc_X$, and prove the membership of x_i in acc_X. However, it should be computationally infeasible to find a witness for any non-accumulated value $y \notin X$ (collision freeness). The dynamic accumulator allows users to add/delete values to a given accumulator and update existing witnesses (without having to completely recalculate these values each time you change the cumulative set). In addition to providing member witnesses, the accumulator should also need to provide non-member witnesses for $y \notin X$, which results in a generic accumulator. Here, collision freeness also covers that it is computationally infeasible to create non-membership witnesses for values $x_i \in X$. Over time, further security properties, that is, undeniability and indistinguishability have been proposed. Undeniability says that it should be computationally infeasible to compute two contradicting witnesses for $z \in X$ and $z \notin X$. Indistinguishability says that neither the accumulator nor the witnesses leak information about the accumulated set X.

Based on different number theory assumptions, many accumulator schemes with different characteristics have been proposed. Basically, there are two main accumulator schemes: an RSA accumulator based on a strong RSA assumption and a bilinear pair accumulator based on the t-Strong Diffie-Hellman (SDH) assumption.

RSA accumulators: Baric and Pfitzmann improved the accumulator scheme proposed by Benaloh and enhanced the original security notion to collision freeness. Sander proposed using RSA moduli with unknown factorization to construct trapdoor-free accumulators. Camenisch and Lysyanskaya extended Baric's solution to dynamically add/remove values to the accumulator, thus forming the first dynamic accumulator scheme. Their program also supports public updates to existing witnesses, i.e., updates without knowing any trapdoors. Later, Li et al. added support for non-member witnesses in Camenisch, resulting in a universal dynamic accumulator. Lipmaa generalizes the RSA accumulator to the modulus on the Euclidean ring [37].

Bilinear Map Accumulator: Nguyen [38] proposes a dynamic accumulator scheme, the bilinear map accumulator, which is suitable for pairwise friendly groups of prime orders p. It is safe under the t-SDH assumption and allows a maximum of t values to be accumulated from the domain $\mathbb{Z}_p$. Later, Damgard et al. [39] extended Nguyen's scheme to make the bilinear map accumulators have universal features.

The accumulator is initially used for timestamping, which is to record the existence of a value at a particular point in time. Over time, accumulators have become more widely used, such as membership testing, distributed signatures, reliable certificate management, and authenticated dictionaries. An accumulator scheme that allows witnesses to have undisclosed value in zero knowledge is now widely used to revoke group signatures and anonymous credentials. The accumulator is also used for zero coins [40], which is an anonymous extension of the bitcoin cryptocurrency.

3 PRELIMINARIES

In this section, we briefly describe the properties of the bilinear pairing and the q-SDH assumption, which was used in the bilinear map accumulator. We also show the basic algorithm for bilinear pair accumulators. Then, we present a TIT that is required to support dynamic operations. Finally, we discussed the security requirements that the data integrity check scheme should satisfy.

3.1 Bilinear Mapping and q-SDH Assumption

Bilinear mapping (pairing) is a mapping $e : G_1 \times G_2 \rightarrow G_T$, where G_1, G_2, and G_T are cyclic groups of prime order p. Let g_1 and g_2 generate G_1 and G_2, respectively. We require e to be able to calculate efficiently and meet the following conditions:

- Bilinearity: $\forall a, b \in Z_p$ $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$.
- Non-degeneracy: $e(g_1, g_2) \neq 1_{G_T}$. That is, $e(g_1, g_2)$ generates G_T.
- Computability: For all $a, b \in Z_p$, there is a valid algorithm to calculate $e(g_1^a, g_2^b)$.

We define a Bilinear Pairing Instance Generator as a PPT algorithm $\mathcal{BPG}$ that takes as input a security parameter 1^λ and returns a uniformly random tuple $t = (p, G_1, G_2, G_T, e, g_1, g_2)$ of bilinear pairing parameters, including a prime number p of size λ, two cyclic groups G_1 and G_2 of order p, a multiplicative group G_T of order p, a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ and two generator g_1 of G_1, g_2 of G_2.

q-SDH assumption: Let p be a prime number with a bit length of κ, G be a finite cyclic group of order p, g be a generator of G, $\alpha \in_R Z_p^*$; it means that α is randomly selected from Z_p^*. For PPT algorithm $\mathcal{A}$, the following function is negligible in κ:

$$\Pr\left[(c, g^{\frac{1}{\alpha+c}}) \leftarrow \mathcal{A}(g, g^{\alpha}, \ldots\ldots, g^{\alpha^q})\right] \leq \epsilon(\kappa) for some \mathrm{\ c} \in Z_p \backslash \{-\alpha\}$$

where Pr[] denotes the probability that the result is true after executing the Procedures, and $\epsilon(\kappa)$ represents a negligible function.

3.2 The Basic Algorithm of Bilinear-Map Accumulator

The encryption accumulator scheme allows the finite set $X = \{x_1, \ldots, x_n\}$ to be accumulated into the compact value acc_X, which is the so-called accumulator. For each element $x_i \in X$, the so-called witness wit_{x_i} can be effectively calculated to prove the membership of x_i in acc_X. The bilinear pair accumulator is based on the q-SDH assumption and works as follows:

(1) $Gen(1^k, t)$: Input the security parameter κ and select two groups G and G_T with prime order p to generate a bilinear pair $e: G \times G \rightarrow G_T$. Among them, the generator of G is g, and $\mathrm{s} \overset{R}{\leftarrow} Z_p^*$ is selected at the same time. Finally, the algorithm returns a set of keys $(sk_{acc}, pk_{acc}) \leftarrow (s, (g, g^s, \ldots\ldots, g^{s^t}))$.

(2) $Eval_r((sk_{acc}, pk_{acc}), X)$: Input the key pair (sk_{acc}, pk_{acc}) and the set $X = \{x_1, \ldots, x_n\}$. If sk_{acc} exists, the accumulated value of set X can be calculated as follows:

$$acc_X = g^{\prod_{i=1}^{n}(x_i+s)}$$

If sk_{acc} is unknown, the accumulated value of set X cannot be directly calculated. So, calculate $\prod_{x\in X}(x+s)$ and denote it as $\sum_{i=0}^{n} a_i \cdot s^i$, the accumulated value of the set X can be calculated as follows:

$$acc_X = \prod_{i=0}^{n} (g^{s^i})^{a_i}$$

Finally, the algorithm returns the accumulated value acc_X and the auxiliary value $aux = (X)$.

(3) $WitCreate((sk_{acc}, pk_{acc}), acc_X, aux, x_i)$: Input the key pair (sk_{acc}, pk_{acc}), the accumulated value acc_X, the auxiliary value $aux = (X)$, and the element x_i. The algorithm first verifies whether the element x_i belongs to the set X. If not, return $\perp$. If sk_{acc} is present, the witness of element x_i can be calculated as follows:

$$wit_{x_i} = acc_X{}^{(x_i+s)^{-1}}$$

If sk_{acc} is unknown, wit_{x_i} cannot be calculated directly. But we can calculate $\prod_{x\in X\backslash\{x_i\}}(x+s)$ and denote it as $\sum_{i=0}^{n-1} a_i \cdot s^i$, then the witness of the element x_i can be calculated as follows:

$$wit_{x_i} = \prod_{i=0}^{n-1} (g^{s^i})^{a_i}$$

(4) $Verify(pk_{acc}, acc_X, wit_{x_i}, x_i)$: The algorithm determines whether the element x_i is a member of the set X by verifying $e(acc_X, g) = e(wit_{x_i}, g^{x_i} g^s)$. If the above formula holds, the algorithm returns true; otherwise, it returns false.

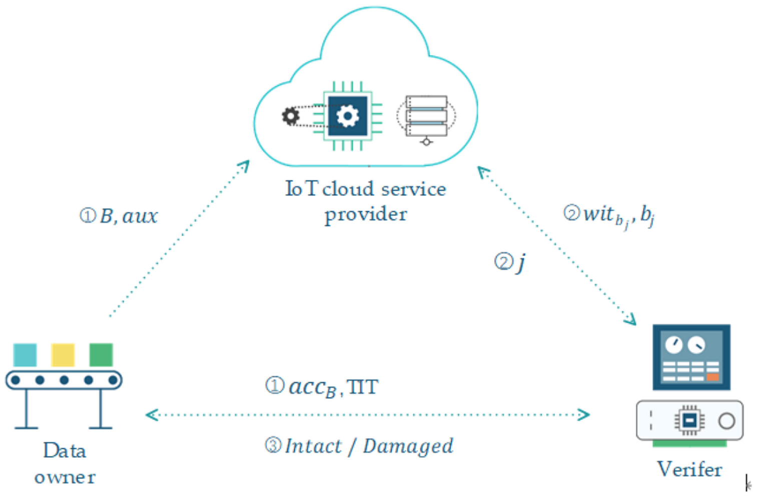

Fig. 2. Accumulator scheme framework.

3.3 Tag Index Table

The TIT is obtained by a certain simplified modification according to the map version table [41]. Data owners implement dynamic data manipulation by generating tags τ_i that are unique to each block of data and storing them on the verifier. The data structure is mainly composed of the following two parts: the first part is the index of the data block, by which the specific position of the data block can be quickly located; the second part is the label value corresponding to each data block, and the value is anti-collision. The hash algorithm is calculated. When performing data integrity verification, the verifier can quickly locate the challenge block through the TIT, thereby efficiently verifying whether the certificate of the cloud storage provider response corresponds. Meanwhile, the data owner can quickly perform data insertion, deletion, and updates through the TIT.

4 DATA INTEGRITY VERIFICATION SCHEME BASED ON BILINEAR MAP ACCUMULATOR

4.1 Model of the Proposed Scheme

Under normal circumstances, any cloud data integrity verification scheme includes the following main roles as shown in Figure 2:

(1) Cloud service provider (CSP): The CSP stores outsourced data and provides management services for data owners;
(2) Data owner: The data owner outsources all of the local data to the cloud storage provider, while not saving a copy of the outsourced data locally;
(3) Verifier: The verifier provides the data owner with integrity verification of the outsourced data.

This scheme can be applied to the edge computing framework, the three roles in the cloud data integrity verification scheme can match the roles in the edge computing framework. Among them, the IoT cloud storage data center and CSP functions are basically the same, the sensor nodes [42, 43] in the wireless sensor network act as data owners, and the edges [44] act as verifiers. We design a scheme, which is based on the bilinear-map accumulator, meeting the characteristics and security requirements mentioned in Sections 5 and 6.

The scheme divides the data M into n segments, each segment having l_1 bits, i.e., $M = \{m_1, \dots m_n\}$. The setup phase generates an l_2 bit flag τ_i for each segment m_i and stores it in the TIT. The data owner generates a processed data block b_i through the tag and the data segment corresponding to the tag, such that $B = \{b_1, \dots b_n\}$. The data owner uses the bilinear pair accumulator to accumulate the processed data B to generate a short value, and then outsources the dataset B and the corresponding auxiliary information to the IoT cloud service provider. To verify the integrity of M, the verifier uses the random data block index j to challenge the IoT cloud service provider. After receiving the challenge, the IoT cloud service provider locates the data block b_j according to the index j, and then accumulates all the other data blocks except the challenge block b_j by using the bilinear pairing accumulator to calculate the witness wit_{b_j} of the block b_j. Note that $ProofGen$ forces the IoT cloud service provider to use the elements in B other than b_j to calculate the witness wit_{b_j} for block b_j. The IoT cloud service provider then returns wit_{b_j} and b_j to the authenticator, which the verifier uses to perform $Verify$ to check data integrity. The program consists of the following components.

4.2 The Basic Data Integrity Verification Scheme

In this section, we will present in detail the basic data integrity verification scheme based on bilinear pair accumulators. In this scheme, the data outsourced to the cloud storage provider is static. The basic solution we proposed consists of the following phases: the setup phase, the challenge and the proof generation phase, and the verification phase.

4.2.1 Setup Phase. Before the data owner outsources the data to the IoT cloud service provider, the outsourced data needs to be initialized. The data owner generates an instance of the accumulator by entering a security parameter λ.: Use $\mathcal{BPG}$ algorithm to generate a tuple $t = (p, G_1, G_2, G_T, e, g_1, g_2)$ and s $\overset{R}{\leftarrow} Z_p^*$. Then, compute a tuple $t' = (g_2, g_2^s, \dots\dots, g_2^{s^n})$, where n is the upper bound on the number of elements to be accumulated by the accumulator. The data owner divides the original data file M into n segments with each segment having l_1 bits, i.e., $\text{M} = \{m_1, \dots m_n\}$. The data owner encrypts each segment separately using symmetric encryption techniques such that $c_i = E(m_i)$. The data owner generates a flag τ_i of l_2 bits for each segment (m_i or c_i) such that $\tau_i = H(m_i)$ or $\tau_i = H(c_i)$. The data owner saves copy τ_i^s and index of each data block in the local TIT and then stores the table on the verifier side for data integrity verification. The data owner combines the calculated tag τ_i with the corresponding data segment m_i or c_i to generate a processed data block b_i that can be calculated, i.e., $b_i = m_i \| \tau_i$ or $b_i = c_i \| \tau_i$ such that $B = \{b_1, \dots b_n\}$.

The data owner calculates the cumulative value of the data file $acc_B = g_1^{\prod_{i=1}^{n}(b_i+s)}$, and stores the auxiliary value $aux_1 = (acc_B, e, g_1, g_2)$ to the verifier. Meanwhile, the data owner outsources the processed data $B = \{b_1, \dots b_n\}$ and the auxiliary value $aux_2 = (g_2, g_2^s, \dots\dots, g_2^{s^n})$ to the IoT cloud service provider. Then, the TIT is stored in the authenticator.

Setup algorithms include:

(1) Input a security parameter λ; generate an instance of the accumulator. Use $\mathcal{BPG}$ algorithm to generate a tuple $t = (p, G_1, G_2, G_T, e, g_1, g_2)$ and s $\overset{R}{\leftarrow} Z_p^*$. Compute a tuple $t' = (g_2, g_2^s, \dots\dots, g_2^{s^n})$, where n is the upper bound on the number of elements to be accumulated by the accumulator.
(2) The data owner divides the original data file M into n segments having l_1 bits, that is, $M = \{m_1, \ \dots \ m_n\}$.
(3) The data owner encrypts each segment separately using symmetric encryption techniques such that $c_i = E(m_i)$.

(4) The data owner generates an l_2 bit flag τ_i for each segment (m_i or c_i) such that $\tau_i = H(m_i)$ or $\tau_i = H(c_i)$, where i is the segment index.
(5) The data owner saves the copy of each tag τ_i^s in the TIT and stores the TIT on the verifier.
(6) The data owner appends each marker τ_i to its corresponding segment (m_i or c_i), and generates a data block $b_i = m_i \| \tau_i$ or $b_i = c_i \| \tau_i$ such that B = $\{b_1, \ldots b_n\}$.
(7) The data owner calculates the cumulative value of the data file B = $\{b_1, \ldots b_n\}$: $acc_B = g_1^{\prod_{i=1}^{n}(b_i+s)}$. The data owner stores the auxiliary value $aux_1 = (acc_B, e, g_1, g_2)$ to the verifier.
(8) The data owner outsources the processed data B = $\{b_1, \ldots b_n\}$ and the auxiliary value $aux_2 = (g_2, g_2^s, \ldots\ldots, g_2^{s^n})$ to the IoT cloud service provider.

4.2.2 Setup Phase. At this stage, the verification asks the IoT cloud service provider to prove the integrity of the outsourced data it has. The verifier challenges the IoT cloud service provider by sending a random block index j. After receiving the challenge, the IoT cloud service provider accumulates all data blocks B = $\{b_1, \ldots b_{j-1}, b_{j+1}, \ldots b_n\}$ except b_j according to the random block index j. Calculate the witness of the challenge block wit_{b_j}.

Since the data owner outsourced auxiliary value aux_2 does not have sk_{acc} = s, the IoT cloud service provider cannot directly calculate the witness of the challenge block $wit_{b_j} = acc_B{}^{(b_i+s)^{-1}}$ without accumulating all data blocks except b_j. Therefore, the IoT cloud service provider can only calculate the witness by the auxiliary value $aux_2 = (g_2, g_2^s, \ldots\ldots, g_2^{s^n})$. The IoT cloud service provider needs to represent the exponent as a polynomial of s, that is, calculate $f(s) = \prod_{b \in B \setminus \{b_i\}}(b+s)$ and denote it as $f(s) = \sum_{i=0}^{n-1} a_i \cdot s^i$, where s represents an unknown number, and $\{a_0, \ldots, a_{n-1}\}$ is a coefficient of s. Therefore, the witness of the challenge block b_j can be calculated as follows:

$$wit_{b_j} = g_2^{a_0} * (g_2^s)^{a_1} * \ldots * (g_2^{s^{n-1}})^{a_{n-1}} = \prod_{i=0}^{n-1} (g_2^{s^i})^{a^i}$$

This allows the IoT cloud service provider to calculate the witness (wit_{b_j}, b_j) when using other blocks than b_j. Next, the IoT cloud service provider sends wit_{b_j} as a proof to the verifier.

Challenge algorithms include:

(1) The verifier uses the random index j to challenge the IoT cloud service provider.

ProofGen algorithms include:

(1) The IoT cloud service provider expresses the exponent as a polynomial of s by $aux_2 = (g_2, g_2^s, \ldots\ldots, g_2^{s^n})$, i.e., calculates $f(s) = \prod_{b \in B \setminus \{b_i\}}(b+s)$ and denotes it as $f(s) = \sum_{i=0}^{n-1} a_i \cdot s^i$, where s represents an unknown number, and $\{a_0, \ldots, a_{n-1}\}$ is the coefficient of s○
(2) The witness of the IoT cloud service provider calculates the challenge block: $wit_{b_j} = \prod_{i=0}^{n-1} (g_2^{s^i})^{a^i}$ ○
(3) The IoT cloud service provider sends (wit_{b_j}, b_j) as a certificate to the authenticator.

4.2.3 Setup Phase. After receiving (wit_{b_j}, b_j), the verifier verifies $e(acc_B, g_2) \overset{?}{=} e(g_1^{b_j} g_1^s, wit_{b_j})$. If the above formula is established, proceed to the next step. As can be seen from the setup phase, the challenge block b_j is composed of the encrypted data block c_i and the segment flag τ_i, and the flag τ_i is generated by $\tau_i = H(c_i)$. The verifier extracts the data segment c_j and its corresponding flag τ_j from the challenge block b_j, and calculates $\tau_j' = H(c_i)$ using the encrypted data segment c_j. The verifier compares whether the calculated flag τ_j' and the extracted flag τ_j are equal, and if they are equal, proceeds to the next step.

$Verify$ algorithms include:

(1) After receiving (wit_{b_j}, b_j), the verifier verifies $e(acc_B, g_2) \overset{?}{=} e(g_1^{b_j} g_1^s, wit_{b_j})$.
(2) The verifier extracts the encrypted data segment c_j and the corresponding flag τ_j from the challenge block b_j, and uses the encrypted data segment c_j to calculate the corresponding flag $\tau_j' = H(c_i)$.

The verifier uses the TIT to locate the tag τ_j^s of the challenge block and verifies $\tau_j' \overset{?}{=} \tau_j$.

4.3 Dynamic Data Support

In the actual application situation, the data that the data owner outsources to the IoT cloud service provider is dynamic, and the data owner can insert, delete, and update the data stored in the cloud at any time. Therefore, our proposed solution also supports dynamic data operations. The specific steps are as follows.

4.3.1 Data Insert Operation. When the data owner needs to insert a new data block $\{m_h, \ldots m_{h+z}\}$ into the data stored in the cloud, the specific steps are as follows:

(1) The data owner encrypts each new data block $\{m_h, \ldots m_{h+z}\}$ separately using symmetric encryption techniques such that $c_i = E(m_i)$. The data owner generates a flag τ_i of l_2 bits for the new data segment $\{m_h, \ldots m_{h+z}\}$ such that $\tau_i = H(c_i), \{\tau_h, \ldots \tau_{h+z}\}$, where $h \leq i \leq h + z$.
(2) The data owner appends each tag τ_i to its corresponding segment, generating a new data block $\{b_h, \ldots b_{h+z}\}$ to be inserted.
(3) The data owner sends an insert request to the IoT cloud service provider, the request being made by the data block to be inserted and its index $\{(b_h, h), \ldots\ldots (b_{h+z}, h + z)\}$.
(4) The IoT cloud service provider sends an insertion confirmation message to the data owner.
(5) When the data owner receives the insertion confirmation message, it adds a new line to the appropriate location in the TIT for each new data block $\{b_h, \ldots b_{h+z}\}$.
(6) The data owner updates the accumulated value acc_B and the auxiliary value aux_2 so that $acc_B^{'} = acc_B^{\prod_{i=h}^{h+z}(b_i+s)}$ and $aux_2 = (g_2, g_2^s, \ldots\ldots, g_2^{s^n}, g_2^{s^{n+h}}, \ldots, g_2^{s^{n+h+z}})$.
(7) The data owner stores the updated accumulated value $acc_B^{'}$ and TIT on the verifier, and the updated auxiliary value $aux_2 = (g_2, g_2^s, \ldots\ldots, g_2^{s^n}, g_2^{s^{n+h}}, \ldots, g_2^{s^{n+h+z}})$ is sent to the IoT cloud service provider.

4.3.2 Data Delete Operation. When the data owner needs to delete a portion of data block $\{b_h, \ldots b_{h+z}\}$ into the data stored in the cloud, the specific steps are as follows:

(1) The data owner uses the tag index table to quickly locate the data block that needs to be deleted, and then sends the index of the data block $\{h, \ldots\ldots, h + z\}$ that needs to be deleted as a delete request to the IoT cloud service provider.
(2) The IoT cloud service provider uses the received index to locate and delete the data block $\{b_h, \ldots b_{h+z}\}$, and then sends a delete confirmation message to the data owner.
(3) After receiving the message, the data owner deletes the corresponding data segment from the TIT.
(4) The data block $\{b_h, \ldots b_{h+z}\}$ is deleted, and the data owner updates the accumulated value acc_B and the auxiliary value aux_2. Finally, get $acc_B' = acc_B^{(\prod_{i=h}^{h+z}(b_i+s))^{-1}}$, $aux_2 = (g_2, g_2^s, \ldots, g_2^{s^{h-1}}, \ldots, g_2^{s^{h+z+1}}, \ldots, g_2^{s^n})$.

(5) The data owner stores the updated accumulated value acc'_B and TIT on the verifier, sending the updated auxiliary value $aux_2 = (g_2, g_2^s, \ldots, g_2^{s^{h-1}}, \ldots, g_2^{s^{h+z+1}}, \ldots, g_2^{s^n})$ to the IoT cloud service provider.

4.3.3 Data Update Operation. The data owner's update operation is more complicated than the insert and delete operations. The data owner needs to delete the current data block first, and then add the data block $\{b'_h, \ldots, b'_{h+z}\}$ that needs to be updated. The specific steps are as follows:

(1) The data owner sends an update request to the IoT cloud service provider, which consists of the updated data block and its index $\{(b'_h, h), \ldots\ldots (b'_{h+z}, h+z)\}$.
(2) After receiving the update request, the IoT cloud service provider locates the data block that needs to be updated according to the received index $\{h, \ldots\ldots, h+z\}$.
(3) The IoT cloud service provider replaces the specified location data block with the data block $\{b'_h, \ldots, b'_{h+z}\}$ in the update request and sends an update confirmation message to the data owner.
(4) After receiving the message, the data owner updates the accumulated value acc_B, so that $acc'_B = acc_B^{(\prod_{i=h}^{h+z}(b_i+s))^{-1} \cdot \prod_{i=h}^{h+z}(b'_i+s)}$.
(5) The data owner stores the updated accumulated value acc'_B on the verifier. Since the total amount of data blocks does not change, the TIT and the auxiliary value aux do not need to be updated.

5 CHARACTERISTICS OF THE CLOUD DATA INTEGRITY SCHEME

Our proposed solution satisfies all the properties of the data integrity scheme [45]. The specific implementations of these characteristics are as follows.

5.1 Remote Verification

In Section 4.2.3, the verifier uses the index j of the random data block to challenge the IoT cloud service provider when performing data integrity verification. After receiving the challenge, the IoT cloud service provider locates the target block b_j according to the index of the data block, and then uses the bilinear pair accumulator to calculate the corresponding witness wit_{b_j}. Finally, the IoT cloud service provider returns the target block b_j and the witness wit_{b_j} as a response to the verifier. Therefore, our solution can implement remote verification.

5.2 Unrestricted Challenge Frequency

Although the number of times that our proposed solution can be challenged for data integrity verification seems to be limited to the number of data blocks, because the verifier uses a random block index when challenging the cloud service provider, this approach allows the verifier to challenge the IoT cloud service provider indefinitely.

5.3 Soundness

In the proposed scheme, the cloud service provider's behavior of caching the previously calculated witnesses in response to the new challenge from the verifier has no specific significance. When the verifier performs data integrity verification to challenge the cloud service provider, the cloud service provider needs to return the calculated witness together with the target data block as a response to the verifier. Since the cached witness needs to store both the target block and the corresponding witness, the IoT cloud service provider needs to provide more storage space. Furthermore, since the data owner outsourced auxiliary value aux_2 does not have sk_{acc} = s, the IoT cloud service provider cannot directly calculate the witness of the challenge block wit_{b_j} without accumulating all data blocks except b_j. Therefore, the IoT cloud service provider can only calculate the witness by the auxiliary value $aux_2 = (g_2, g_2^s, \ldots\ldots, g_2^{s^n})$. If the IoT cloud service provider does not actually save the data or the data is corrupted, the valid witness cannot be correctly calculated.

5.4 Stateless Verification

Our proposed scheme satisfies the feature of stateless verification. Since the verifier uses a random data block index j as a challenge to send to the IoT cloud service provider, the new challenge request from verification to IoT cloud service provider is independent of all past data integrity verification. The IoT cloud service provider locates the target block b_j based on the challenge and calculates the corresponding witness wit_{b_j}. According to Section 5.3, it is meaningless for the IoT cloud service provider to cache the behavior of witnesses calculated in the past, so every calculated witness is unique.

5.5 Robustness

According to Section 4.2.1, the data owner first divides the original file into n data blocks, and then generates processed data blocks B by using encrypted data blocks and generated tags. IoT cloud service providers need to accumulate all data blocks except the target block when calculating witnesses, so even a trivial change to the data outsourced to the IoT cloud service provider can cause the generated witness to change. The IoT cloud service provider knows that the verifier performs verification $e(acc_B, g_2) \overset{?}{=} e(g_1^{b_j} g_1^s, wit_{b_j})$ by proof (wit_{b_j}, b_j) and the accumulated value acc_B stored in the verifier. This is due to the verifier extracting the encrypted data segment c_j and the corresponding flag τ_j from the challenge block b_j, and using the encrypted data segment c_j to calculate the corresponding flag $\tau_j' = H(c_i)$. It is determined whether the data block in the response returned by the IoT cloud service provider is the target block by comparing whether τ_j' and τ_j are the same. Therefore, the IoT cloud service provider cannot change the challenge block b_j.

5.6 Data Recovery

The verifier performs verification $e(acc_B, g_2) \overset{?}{=} e(g_1^{b_j} g_1^s, wit_{b_j})$ by proof (wit_{b_j}, b_j) and the accumulated value acc_B stored in the verifier. So even a trivial change to the data outsourced to the IoT cloud service provider can cause the generated witness to change. To support data recovery and error detection, the solutions based on error correction codes can be easily integrated into our solution.

5.7 Dynamic Data Handling

In the real world, data owners often need to perform dynamic operations on data that is outsourced to the IoT cloud service provider, i.e., data stored in the cloud can be inserted, deleted, and updated at any time. As proposed in Section 4.3, our proposed solution implements dynamic data operations by using a TIT, and the specific steps of the data integrity verification scheme remain unchanged when performing dynamic operations.

5.8 Public/Private Auditability

Our proposed approach meets both public and private auditability. In our scheme, the data owner can perform data integrity verification while also allowing the third-party auditor to perform the verification process without retrieving the remote data. The steps for the data owner to perform the verification process are negligible. Data owners use symmetric encryption to encrypt data to prevent data leakage such that $c_i = E(m_i)$. When a third-party auditor performs data integrity verification, the specific steps are as follows:

(1) The data owner sends the signature request with the private key PK_{DO} and its certificate $Cert_{DO}$ to the third-party auditor: $R_1: \ (E_{PK_{DO}}(ID_{DO}), Cert_{D0})$.
(2) The third-party auditor verifies the signature of the data owner and sends the identity data of the data owner and third-party auditor encrypted with the private key PK_{TPA} and the certificate of the third-party auditor to the data owner in response: $R_2: \ (E_{PK_{TPA}}(ID_{DO}, ID_{TPA}), Cert_{TPA})$.
(3) The data owner sends a message to the cloud service provider, which consists of the identity data of the data owner ID_{DO} and third-party auditor ID_{TPA}, the certificate of the third-party auditor $Cert_{TPA}$, and the private key of the data owner PK_{DO}: $R_3: (E_{PK_{DO}}(ID_{DO}, ID_{TPA}, Cert_{TPA}))$.
(4) After receiving the message, the cloud service provider verifies the received information and stores the received message in the cloud service provider's database.
(5) When a third-party auditor is ready for data integrity verification, the third-party auditor needs to send a permission verification request consisting of the identity data of the data owner ID_{DO} and the third-party auditor ID_{TPA}, and the certificate of the third-party auditor to the cloud service provider $Cert_{TPA}$.
(6) The cloud service provider verifies the identity data and the certificate in the received request and the information in the message stored in the local database. If the verification is passed, the permission verification response is sent to the third-party auditor; if the verification fails, it sends the rejection verification response to the third-party auditor.

5.9 Privacy-preserving

In Section 4.2.1, the data owner encrypts the original data segment using symmetric encryption techniques such that $c_i = E(m_i)$. The data owner separately saves the key of the encrypted data and does not disclose it to other third parties, so no one can access the data saved to the cloud service provider except the data owner himself.

5.10 Fairness

Our data integrity verification solution provides protection for honest cloud service providers, preventing dishonest data owners from accusing honest cloud service providers of manipulating data.

(1) The data owner sends a request to the certificate authority to generate a shared certificate $Cert_{DO,TPA}$. The certificate authority generates a shared public/private key pair $(PU_{DO,TPA}, PR_{DO,TPA})$ between the data owner and the third party auditor: $R_4: (R_2, E_{PK_{DO}}(ID_{DO}, ID_{TPA}), Cert_{DO})$.
(2) According to the X.509 standard [46], the certificate authority uses ID_{DO} and ID_{TPA} as the sole issuer identifier to generate $Cert_{DO,TPA}$.
(3) If the verification process is delegated to the third-party auditor, the data owner encrypts $PU_{DO,TPA}$ and $PR_{DO,TPA}$ and signs,then send it to the third-party auditor:
$R_5: \ E_{PU_{TPA}}(E_{PR_{DO}}(ID_{DO}, ID_{TPA}, PU_{DO,TPA}, PR_{DO,TPA}))$
(4) The third-party auditor verifies the signature and obtains $PU_{DO,TPA}$ and $PR_{DO,TPA}$.
(5) The data owner uses the shared private key $PR_{DO,TPA}$ to generate a l_2 bit flag τ_i for each segment, such that$\tau_i = H(m_i \| PR_{DO,TPA})$.
(6) If a dishonest data owner accuses a cloud service provider of making changes to a block of data b_m, the regulator enforces the third-party auditor to display the shared private key $PR_{DO,TPA}$ based on the key compromise method

(7) The regulator encrypts a random string using the shared public key $PU_{DO,TPA}$ and sends the encrypted data to the third-party auditor. The third-party auditor decrypts the encrypted data using the shared private key $PR_{DO,TPA}$ and sends the decrypted data to the regulatory authority. The regulator verifies that the string has changed to ensure the accuracy of the shared private key $PR_{DO,TPA}$.
(8) The regulator forces the cloud service provider to display data block b_m that are accused of tampering. The regulator extracts the encrypted data segment c_m and the corresponding flag τ_m from the challenge block b_m, and uses shared private key $PR_{DO,TPA}$ and the encrypted data segment c_m to calculate the corresponding flag $\tau'_m = H(c_m \| PR_{DO,TPA})$.
(9) The regulator verified $\tau'_j \overset{?}{=} \tau_j$, and if the verification passed, the cloud service provider did not manipulate the data.

6 SECURITY ANALYSIS

Data integrity verification schemes are exposed to a variety of attacks. We will show how our proposed solution can defend against these possible attacks:

Tag Forgery Attack: The data owner uses the hash algorithm H to generate tags $\tau'_j = H(c'_j)$. Due to the collision-resistance nature of the hash algorithm, the possibility of using other data blocks to generate the same tags is almost negligible. So IoT cloud service providers can't fake tags.

Data Deletion Attack: If the IoT cloud service provider deletes and destroys the original data, it cannot calculate the witness of the challenge block:

$$wit_{b_j} = g_2^{a_0} * (g_2^s)^{a_1} * \ldots * (g_2^{s^{n-1}})^{a_{n-1}} = \prod_{i=0}^{n-1} (g_2^{s^i})^{a^i}$$

when it receives the verifier question. Since each data block b_i is a combination of the tag τ_i and the encrypted data block c_i, the IoT cloud service provider cannot generate valid proof of ownership by using only the tag without actually saving the original data (wit_{b_j}, b_j).

Replacement Attack: If the IoT cloud service provider replaces the corrupted or deleted data with other valid data blocks and tags when the verifier challenges the IoT cloud service provider with the random block index j, then the token τ'_j calculated by the verifier cannot match the extracted token τ_j.

Replay Attack: As described in Section 5.3, it does not make sense for a cloud service provider to cache witnesses from past calculations in response to new challenges from current validators. On the one hand, cloud service provider cache witnesses need to store both the target block and the corresponding witness, which will greatly increase the storage overhead of the cloud service provider; on the other hand, the verifier uses random data block index j to challenge the cloud storage provider when performing data integrity verification, so the probability of the same challenge is basically negligible.

Data Breach Attack: During the setup phase, the data owner encrypts each segment separately using symmetric encryption techniques such that $c_i = E(m_i)$. So, others can't know the data that is outsourced to the IoT cloud service provider.

7 APPLICATION AND PERFORMANCE

7.1 Application

Our proposed scheme can be applied to edge computing, and we can match the three roles in the edge computing framework to the roles in the proposed solution. That is to say, the sensor node in the wireless sensor network [47, 48] acts as the data owner, and the edge acts as the verifier, while the IoT cloud storage data center and CSP function are basically the same. And the computing power from the sensor node to the edge to the IoT cloud service provider is gradually enhanced

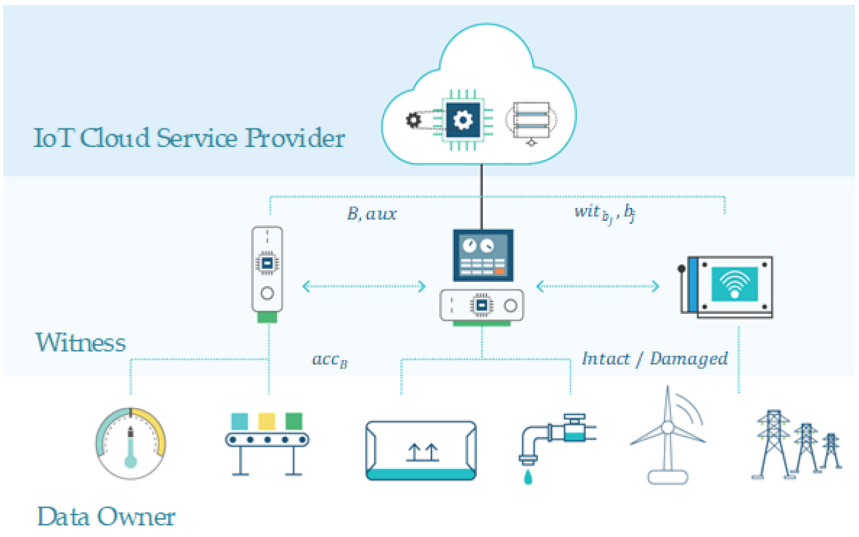

Fig. 3. Accumulator scheme framework.

and can meet the calculation requirements required by the solution. The framework applied to the edge calculation is shown in Figure 3.

In this framework, a variety of sensors collect a large amount of data, but the computing power and storage capacity of the sensor is very limited, so the data is outsourced to the IoT cloud storage provider for storage. However, the data stored in the IoT cloud storage provider may be damaged, tampered with, or even not saved at all. The solution proposed by us can effectively solve such problems. In our solution, it is simple to verify the direction of the IoT cloud storage provider, and only need to question the index of the data block that needs to be challenged, so the requirements for the verifier are lower. Moreover, when the verifier verifies the integrity of the data stored in the IoT cloud storage provider through the proof sent by the IoT cloud storage provider, the computational complexity required for the calculation is constant. So,it can be easily. A device with limited computing power is verified. In the edge computing framework, the edge acts as a verifier and the computational power of the edge is limited, so our solution is well suited for use in it.

7.2 Performance Evaluation

The specific implementation program of the sensor data integrity verification scheme based on the bilinear pair accumulator is written in C++ language. In this scheme, we use the open source C library FLINT for number theory calculations to perform polynomial operations in bilinear pair accumulators. For the bilinear maps and cyclic groups needed to implement the bilinear pair accumulator, we chose the fastest library DCLXVI to calculate the elliptic curve. Our proposed scheme uses the symmetric encryption algorithm AES-128 [49] to encrypt the data blocks that need to be outsourced, and uses the digest generation algorithm SHA-3 [50] to generate conflict-free 160-bit tags. Meanwhile, this scheme is compared with Hao's scheme, using 2048-bit RSA modulus and 8 kb blocks. We set p to a 210-bit prime; when the order p of the bilinear pair is 210-bit, the security strength is equivalent to the 2048-bit RSA modulus. The specific test for this experiment was run on a 64-bit 3.4 GHz Intel Core i7 machine, with 16 GB of RAM and running Sabayon Linux.

In the setup phase, our solution first divides the original data M into n data segments, so in the first step of the experimental test, we must find the optimal size of the data block. We consider the three aspects of the verification side's storage overhead, the cloud service provider's witness

Table 1. Computing and Storage Overhead for Verifiers and Cloud Service Providers Based on Different Block Sizes

Block size (Bytes)	Storage cost (MB)	Proof-gen time (sec)	Verification time
256	163.8	−1.49	−0.93
512	81.9	−1.19	−0.81
768	54.6	−0.67	−0.74
1,024	41.1	−0.08	−0.49
1,280	31.2	0.43	−0.13
1,536	27.6	0.82	0.43
1,792	23.1	1.03	1.01
2,048	19.8	1.17	1.87

Table 2. The Various Costs for Different File Sizes

File size (MB)	Setup time (sec)	Proof-gen time (sec)	Verification time (sec)	Storage cost (MB)
256	2.3	0.4	0.003	6.9
512	6.7	1.7	0.003	14.1
768	11.8	3.1	0.003	20.5
1,024	16.3	4.4	0.003	27.6
1,280	19.4	5.2	0.003	34.2
1,536	22.6	6.3	0.003	41.2
1,792	26.7	7.1	0.003	47.8
2,048	29.1	7.9	0.003	54.6

generation time, and the time required for the verifier to verify. The file we are testing is fixed at 2GB, the base size of the data block is 256 bytes, and the size of each increment is 256 bytes, and the experiment is stopped until the fast size is increased to 2 MB. The final experimental results are shown in Table 1. We used the Z-score normalization method to process the witness generation time of the cloud service provider and the time required for verification by the verifier to make the experimental results clearer. As can be seen from Table 1, we determined that the size of the data block is 768 bytes, which is the best match for our needs.

As can be seen from the experimental results in Table 2, the time required for the data owner's initial setup and the computational overhead required by the cloud service provider to calculate the witness increases proportionally with the increase in the size of the outsourced file. The storage overhead of the verifier is mainly the TIT sent by the data owner. Since the size of the data block has been determined, the larger the outsourcing file is, the more segments are divided and the more tags are generated. Therefore, the verification side's storage overhead also increases proportionally with the increase in the length of the outsourced file. As can be seen from Section 4.2.3, the computational complexity of the verifier in the verification operation remains constant at $O(2)$. Therefore, we can find that the computational overhead of the verifier does not change with the change of the outsourcing file, and its verification time is fixed at 0.003 seconds.

In further experiments, we conducted a comprehensive comparison with Hao's program. The specific experimental results are shown in Table 3. We can see that the performance of our proposed bilinear pairing scheme is very efficient. In Hao's scheme, the amount of exponential computation needed by the verifier to perform integrity verification increases rapidly in proportion to the increase in the length of outsourced data. In our scheme, the amount of calculation

Table 3. Performance of the Proposed Scheme on the 1GB File

Costs	Setup (s)	Challenge Gen (μs)	Proof Gen (s)	Verification (s)	Communication (Bytes)	Storage (MB)
Proposed Scheme	29.1	0.8	7.9	0.003	776	54.6
Hao's Scheme	13,167	4.8	97	59	975	64

required for the verification operation is constant, and its computational complexity is $O(2)$. The time required for verification does not change with the change of the outsourced file. This allows our scheme to run on devices with limited computing power. This means that our proposed solution can be applied to the edge computing framework very well because the computing power of the edge as the verifier is very limited. At the same time, our proposed solution is superior to Hao's solution in the initial setup phase, the verification side's storage overhead and challenge generation, and the cloud service provider's ability to calculate witnesses.

In Khedr's scheme, data integrity verification is performed using the RSA accumulator, and the input domain of the RSA accumulator is limited to prime numbers to avoid collisions. Therefore, the data owner needs to further process the data block. Khedr provides a right shift of each data block to solve this problem, which makes Khedr's solution take more time in the setup phase. Our verification scheme uses a bilinear accumulator, so there is no need to further process the data segment. The cloud service provider uses the RSA accumulator to calculate the witnesses of the target block much more slowly than the bilinear pair accumulator, which means that our proposed scheme is an order of magnitude faster than the Khedr scheme in proof generation. In the data integrity verification phase of the verifier, our solution is slightly faster than Khedr's solution when the size of the outsourced file and the size of the data block are the same.

8 CONCLUSIONS

With the development of 5G, Industry 4.0, Vehicle Interne, Smart Home, and Smart City, more and more devices are connected to the network and realize mutual communication, and the number of IoT devices is developing in a blowout manner. Ensuring massive data integrity is a problem that data owners and cloud storage service providers must solve. In this article, we propose a new and effective sensor data integrity auditing mechanism based on bilinear pair accumulator, which can be well adapted to our edge computing framework. Using our scheme, the verifier verifies that the operation is independent of the number of data blocks during data integrity verification, and its computational complexity is constant. This scheme minimizes the burden and cost of the verification process to the greatest extent, which makes it possible to perform data verification effectively on low-power-consumption devices. Therefore, this scheme is very suitable for the edge computing framework we proposed because the edge computing capacity of the framework as a verifier is very limited. Our solution not only reduces the computing and storage costs of cloud storage providers and validators, but also supports dynamic data operations through the use of tag index table. In terms of security, the scheme can effectively prevent tag forgery, data deletion, replacement, data leakage attack, and replay attack. In future work, we need to further optimize the use of the scheme in reality.

REFERENCES

[1] C. Yin, J. Xi, R. Sun, and J. Wang. 2018. Location privacy protection based on differential privacy strategy for big data in industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3628–3636.

[2] Y. Li, M. Kumar, W. Shi, and J. Wan. 2017. Falcon: An ambient temperature aware thermal control policy for IoT gateways. *Sustainable Computing-Informatics & Systems* 16, 4 (2017), 48–55.

[3] X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K. R. Choo. 2018. A robust and energy efficient authentication protocol for industrial Internet of Things. *IEEE Internet of Things Journal.* 5, 2 (2018), 1606–1615. DOI: https://doi.org/10.1109/JIOT.2017.2787800

[4] G. Jia, G. Han, H. Rao, and L. Shu. 2018. Edge computing-based intelligent manhole cover management system for smart cities. *IEEE Internet of Things Journal* 5, 3 (2018), 1648–1656.

[5] Y. Chen, J. Wang, R. Xia, Q. Zhang, Z. Cao, and K. Yang. 2019. The visual object tracking algorithm research based on adaptive combination kernel. *Journal of Ambient Intelligence and Humanized Computing* 19, 10 (2019), 4855–4867. DOI: https://doi.org/10.1007/s12652-018-01171-4

[6] J. Wang, Y. Gao, W. Liu, A. K. Sangaiah, and H.-J. Kim. 2019. An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *International Journal of Distributed Sensor Networks.* 2019, 3 (2019), 833–847. DOI: https://doi.org/10.1177/1550147719839581

[7] B. Yin and X. We. 2019. Communication-efficient data aggregation tree construction for complex queries in IoT applications. *IEEE Internet of Things Journal* 6, 2 (2019), 3352–3363. DOI: https://doi.org/10.1109/JIOT.2018.2882820

[8] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu. 2017. Network location-aware service recommendation with random walk in cyber-physical systems. *Sensors* 17, 9 (2017), 2059–2071.

[9] Y. J. Ren, Y. Leng, Y. P. Cheng, and J. Wang. 2019. Secure data storage based on blockchain and coding in edge computing. *Mathematical Biosciences and Engineering* 16, 3 (2019), 1874–1892. DOI: https://doi.org/10.3934/mbe.2019091

[10] J. Wang, Y. Gao, W. Liu, W. Wu, and S. Lim. 2019. An asynchronous clustering and mobile data gathering schema based on timer mechanism in wireless sensor networks. *Computer, Materials & Continua* 58, 3 (2019), 711–725.

[11] Y. Ren, Y. Liu, S. Ji, A. K. Sangaiah, and J. Wang. 2018. Incentive mechanism of data storage based on blockchain for wireless sensor networks. *Mobile Information Systems.* 2018, 10 (2018), 158–167. DOI: https://doi.org/10.1155/2018/6874158

[12] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai. 2019. QoS prediction for service recommendation with deep feature learning in edge computing environment. *Mobile Networks and Applications* 25, 4 (2019), 391–401. DOI: https://doi.org/10.1007/s11036-019-01241-7

[13] C. Chen, M. Lin, and C. Liu. 2018. Edge computing gateway of the industrial Internet of Things using multiple collaborative microcontrollers. *IEEE Network* 38, 1 (2018), 24–32.

[14] J. Wang, Y. Gao, X. Yin, F. Li, and H. Kim. 2018. An enhanced PEGASIS algorithm with mobile sink support for wireless sensor networks. *Wireless Communications and Mobile Computing* 2018, 12 (2018) 1–9. DOI: https://doi.org/10.1155/2018/9472075

[15] S. M. H. Rostami, A. K. Sangaiah, J. Wang, and X. Liu. 2019. Obstacle avoidance of mobile robots using modified artificial potential field algorithm. *EURASIP Journal on Wireless Communications and Networking.* 2019, 1(2019), 2075–2085. DOI: https://doi.org/10.1186/s13638-019-1396-2

[16] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei. 2017. Collaborative service selection via ensemble learning in mixed mobile network environments. *Entropy* 19, 7 (2017), 358–375.

[17] A. K. Das, S. Zeadally, and D. He. 2018. Taxonomy and analysis of security protocols for Internet of Things. *Future Generation Computer Systems* 89, 12 (2018), 110–125. DOI: https://doi.org/10.1016/j.future.2018.06.027

[18] X. Li, J. Niu, S. Kumari, F. Wu, A. K. Sangaiah, and K.-K. R. Choo. 2018. A three-factor anonymous authentication scheme for wireless sensor networks in Internet of Things environments. *Journal of Network and Computer Applications* 103, 2 (2018), 194–204. DOI: https://doi.org/10.1016/j.jnca.2017.07.001

[19] J. Pan and J. McElhannon. 2018. Future edge cloud and edge computing for Internet of Things applications. *IEEE Internet of Things Journal* 5, 1 (2018), 439–449.

[20] Y. Ren, Y. Liu, and C. Qian. 2018. Digital continuity guarantee based on data consistency in cloud storage. In *Proceedings of Cloud Computing and Security, Cham* l (2018), 3–11.

[21] Y. J. Ren, Y. Leng, F. J. Zhu, J. Wang, and H-J. Kim. 2019. Data storage mechanism based on blockchain with privacy protection in wireless body area network. *Sensors.* 19, 10 (2019), 2395–2408. DOI: https://doi.org/10.3390/s19102395

[22] Y. Yin, W. Xu, Y. Xu, H. Li, and L. Yu. 2017. Collaborative QoS prediction for mobile service with data filtering and slopeone model. *Mobile Information Systems.* 2017, 6 (2017), 1–14.

[23] Y. J. Ren, F. J. Zhu, J. Qi, J. Wang, and A. K. Sangaiah. 2019. Identity management and access control based on blockchain under edge computing for the industrial Internet of Things. *Applied Sciences* 9, 10 (2019), 2058–2074. DOI: https://doi.org/10.3390/app9102058

[24] E. Hesham, S. Sharmi, P. Mukesh, P. Deepak, G. Akshansh, M. Manoranjan, and C. Lin. 2017. Edge of things: The big picture on the integration of edge, IoT, and the cloud in a distributed computing environment. *IEEE Access* 5, 6 (2017), 1706–1717.

[25] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison. 2017. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective. *IEEE Journal on Selected Areas in Communications* 35, 11 (2017), 2586–2595.

[26] Y. Liu, Y. Ren, C. Ge, J. Xia, and Q. Wang. 2019. A CCA-secure multi-conditional proxy broadcast re-encryption scheme for cloud storage system. *Journal of Information Security and Applications* 47, 8 (2019), 125–131. DOI: https://doi.org/10.1016/j.jisa.2019.05.002

[27] J. Zhang, X. Jin, J. Sun, J. Wang, and A. K. Sangaiah. 2018. Spatial and semantic convolutional features for robust visual object tracking. *Multimedia Tools and Applications* 79, 6 (2020), 15095–15115. DOI: https://doi.org/10.1007/s11042-018-6562-8

[28] R. C. Kim-Kwang, G. Stefanos, H. P. Jong. 2018. Cryptographic solutions for industrial internet-of-things: Research challenges and opportunities. *IEEE Transactions on Industrial Informatics.* 14, 8 (2018), 3567–3569.

[29] G. Caronni and M. Waldvogel. 2003. Establishing trust in distributed storage providers. In *Proceedings of 3rd International Conference on Peer-to-Peer Computing.* 128–133.

[30] Y. Deswarte, J.-J. Quisquater, and A. Saïdane. 2004. Remote integrity checking: Integrity and internal control in information systems VI. 1–11.

[31] D. L. G. Filho and P. S. L. M. Barreto. 2006. Demonstrating data possession and uncheatable data transfer. *IACR Cryptology ePrint Archive.* 2006 (2006), 150.

[32] V. Sebé, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater. 2008. Efficient remote data possession checking in critical information infrastructures. *IEEE Transactions on Knowledge and Data Engineering* 20, 6 (2008), 1034–1038.

[33] A. F. Barsoum and M. A. Hasan. 2010. Provable possession and replication of data over cloud servers. *Centre For Applied Cryptographic Research (CACR), University of Waterloo.* 32.

[34] Z. Hao, S. Zhong, and N. Yu. 2011. A privacy preserving remote data integrity checking protocol with data dynamics and public verifiability. *IEEE Transactions on Knowledge and Data Engineering* 23, 9 (2011), 1432–1437.

[35] W. Khedr1, H. Khater1, and E. Mohamed. 2019. Cryptographic accumulator-based scheme for critical data integrity verification in cloud storage. *IEEE Access* 7, 5 (2019), 65635–65651. DOI: https://doi.org/10.1109/ACCESS.2019.2917628

[36] J. Benaloh and M. de Mare. 1994. One-way accumulators: A decentralized alternative to digital signatures. In *Proceedings of Advances in Cryptology — EUROCRYPT '93, Berlin.* 274–285.

[37] H. Lipmaa. 2012. Secure accumulators from Euclidean rings without trusted setup. In *Proceedings of Applied Cryptography and Network Security, Berlin.* 224–240.

[38] L. Nguyen. 2005. Accumulators from bilinear pairings and applications. In *Proceedings of Topics in Cryptology – CT-RSA 2005, Berlin.* 275–292.

[39] I. Damgård and N. Triandopoulos. 2008. Supporting non-membership proofs with bilinear-map accumulators. *IACR Cryptology ePrint Archive.* 2008 (2008), 538.

[40] I. Miers, C. Garman, M. Green, and A. D. Rubin. 2013. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proceedings of 2013 IEEE Symposium on Security and Privacy.* 397–411.

[41] A. F. Barsoum and M. A. Hasan. 2015. Provable multicopy dynamic data possession in cloud computing systems. *IEEE Transactions on Information Forensics and Security* 10, 3 (2015), 485–497.

[42] J. Wang, X. Gu, W. Liu, A. K. Sangaiah, and H. Kim. 2019. An empower Hamilton loop based data collection algorithm with mobile agent for WSNs. *Human-centric Computing and Information Sciences* 9, 18 (2019), 2659–2672. DOI: https://doi.org/10.1186/s13673-019-0179-4

[43] B. Yin, S. Zhou, S. Zhang, K. Gu, and F. Yu. 2017. On efficient processing of continuous reverse skyline queries in wireless sensor networks. *KSII Transactions on Internet and Information Systems* 11, 4 (2017), 1931–1953.

[44] M. Gusev and S. Dustdar. 2018. Going back to the roots—The evolution of edge computing, an IoT perspective. *IEEE Internet Computing.* 22, 2 (2018), 5–15.

[45] F. Zafar, A. Khan, S. U. R. Malik, M. Ahmed, A. Anjum, and M. I. A. Khan. 2017. Survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers & Security.* 65, 3 (2017), 29–49.

[46] R. Housley, W. Polk, W. Ford, and D. Solo. 2008. Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile. 1721–2070.

[47] J. Wang, C. Ju, Y. Gao, A. K. Sangaiah, and G. Kim. 2018. A PSO based energy efficient coverage control algorithm for wireless sensor networks. *Computers Materials & Continua.* 56, 3 (2018), 433–446.

[48] C. Ge, Z. Liu, J. Xia, and L. Fang. 2019. Revocable identity-based broadcast proxy re-encryption for data sharing in clouds. *IEEE Transactions on Dependable and Secure Computing* 19, 2 (2019), 1–1. DOI: https://doi.org/10.1109/TDSC.2019.2899300

[49] J. Daemen and V. Rijmen. 2013. The design of rijndael: AES-the advanced encryption standard. *Springer Science & Business Media.*

[50] M. J. Dworkin. 2015. Sha-3 standard: Permutation-based hash and extendable-output functions. *Federal Inf. Process. Stds. (NIST FIPS)-202.*

Received October 2019; revised December 2019; accepted January 2020