Data Integrity Auditing without Private Key Storage for Secure Cloud Storage

Wenting Shen, Jing Qin, Jia Yu, Rong Hao, Jiankun Hu, *Senior Member*, *IEEE* and Jixin Ma

***Abstract*—Using cloud storage services, users can store their data in the cloud to avoid the expenditure of local data storage and maintenance. To ensure the integrity of the data stored in the cloud, many data integrity auditing schemes have been proposed. In most, if not all, of the existing schemes, a user needs to employ his private key to generate the data authenticators for realizing the data integrity auditing. Thus, the user has to possess a hardware token (e.g. USB token, smart card) to store his private key and memorize a password to activate this private key. If this hardware token is lost or this password is forgotten, most of the current data integrity auditing schemes would be unable to work. In order to overcome this problem, we propose a new paradigm called data integrity auditing without private key storage and design such a scheme. In this scheme, we use biometric data (e.g. iris scan, fingerprint) as the user's fuzzy private key to avoid using the hardware token. Meanwhile, the scheme can still effectively complete the data integrity auditing. We utilize a linear sketch with coding and error correction processes to confirm the identity of the user. In addition, we design a new signature scheme which not only supports blockless verifiability, but also is compatible with the linear sketch. The security proof and the performance analysis show that our proposed scheme achieves desirable security and efficiency.**

***Index Terms*—Cloud storage, Data integrity auditing, Data security, Biometric data**

I. Introduction

CLOUD storage can provide powerful and on-demand data storage services for users [1]. By using the cloud service, users can outsource their data to the cloud without wasting substantial maintenance expenditure of hardware and software, which brings great benefits to users. However, once the users upload their data to the cloud, they will lose the physical control of their data since they no longer keep their data in local. Thus, the integrity of the cloud data is hard to be guaranteed, due to the inevitable hardware/software failures and human errors in the cloud [2].

Many data integrity auditing schemes have been proposed to allow either the data owner or the Third Party Auditor (TPA) to check whether the data stored in the cloud is intact or not. These schemes focus on different aspects of data integrity auditing, such as data dynamic operation [3–5], the privacy protection of data and user identities [6–8], key exposure resilience [9–11], the simplification of certificate management [12, 13] and privacy-preserving authenticators [14], etc. In the above data integrity auditing schemes, the user needs to generate authenticators for data blocks with his private key. It means that the user has to store and manage his private key in a secure manner [15]. In general, the user needs a portable secure hardware token (e.g. USB token, smart card) to store his private key and memorizes a password that is used to activate this private key. The user might need to remember multiple passwords for different secure applications in practical scenarios, which is not user friendly. In addition, the hardware token that contains the private key might be lost. Once the password is forgotten or the hardware token is lost, the user would no longer be able to generate the authenticator for any new data block. The data integrity auditing will not be functioning as usual. Therefore, it is very interesting and appealing to find a method to realize data integrity auditing without storing the private key.

A feasible method is to use biometric data, such as fingerprint and iris scan [16, 17], as the private key. Biometric data, as a part of human body, can uniquely link the individual and the private key. Unfortunately, biometric data is measured with inevitable noise each time and cannot be reproduced precisely [18] since some factors can affect the change of biometric data. For example, the finger of each person will generate a different fingerprint image every time due to pressure, moisture, presentation angle, dirt, different sensors, and so on. Therefore, the biometric data cannot be used directly as the private key to generate authenticators in data integrity auditing.

Contribution. The contribution of this paper can be summarized as follows:

We initiate the first study on how to employ biometric data as fuzzy private key to perform data integrity auditing, and propose a new paradigm called data integrity auditing without private key storage. In such a scheme, a user utilizes biometric data as his fuzzy private key for confirming his identity. The

This research is supported by National Natural Science Foundation of China (61772311, 61572267, 61272091), National Cryptography Development Fund of China (MMJJ20170118), the Open Project of Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, the Open Project of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences (2017-MS-21, 2017-MS-05). (*Corresponding author: J. Qin*)

W. Shen is with the School of Mathematics, Shandong University, Shandong 250100, China. E-mail:Shenwentingmath@163.com.

J. Qin is with the School of Mathematics, Shandong University, Shandong 250100, China, with State Key Laboratory of Cryptology,P.O. Box 5159,Beijing,10078,China. E-mail:qinjing@sdu.edu.cn.

J. Yu is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China, with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. E-mail:qduyujia@gmail.com.

R. Hao is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China. E-mail:hr@qdu.edu.cn.

J. Hu is with the Chair Professor and Research Director of Cyber Security Lab, School of Engineering and IT, University of New South Wales at the Australian Defence Force Academy (UNSW@ADFA), Canberra, Australia. E-mail:J.Hu@adfa.edu.au.

J. Ma is with is the director and academic-reader of Computing and Mathematical Sciences Department at University of Greenwich, the United Kingdom. E-mail:j.ma@greenwich.ac.uk

data integrity auditing can be performed under the condition that there is not any hardware token for storing the private key. We further formalize the definition of data integrity auditing scheme without private key storage for secure cloud storage.

We design a practical data integrity auditing scheme without private key storage for secure cloud storage. In our scheme, two fuzzy private keys (biometric data) are extracted from the user in the phase of registration and the phase of signature generation. We respectively use these two fuzzy private keys to generate two linear sketches that contain coding and error correction processes. In order to confirm the user's identity, we compare these two fuzzy private keys by removing the "noise" from two sketches. If the two biometric data are sufficiently close, we can confirm that they are extracted from the same user; otherwise, from different users. How to design a signature satisfying both the compatibility with the linear sketch and the blockless verifiability is a key challenge for realizing data integrity auditing without private key storage. In order to overcome this challenge, we design a new signature scheme named as MBLSS by modifying the BLS short signature based on the idea of fuzzy signature. We give the security analysis and justify the performance via concrete implementations. The results show that the proposed scheme is secure and efficient.

A. *Related Work*

Ateniese et al. [19] firstly proposed the notion of Provable Data Possession (PDP). They employed the random sample technique and homomorphic linear authenticators to design a PDP scheme, which allows an auditor to verify the integrity of cloud data without downloading the whole data from the cloud. Juels and Kaliski [20] proposed the concept of Proof of Retrievability (PoR). In the proposed scheme, the error-correcting codes and the spot-checking technique are utilized to ensure the retrievability and the integrity of the data stored in the cloud. Shacham and Waters [21] constructed two PoR schemes with private verifiability and public verifiability by using pseudorandom function and BLS signature.

To support user-interactions, including data modification, insertion and deletion, Zhu et al. [22] constructed a dynamic data integrity auditing scheme by exploiting the index hash tables. Sookhak et al. [23] also considered the problem of data dynamics in data integrity auditing and designed a data integrity auditing scheme supporting data dynamic operations based on the Divide and Conquer Table. In public data integrity auditing, the TPA might derive the contents of user's data by challenging the same data blocks multiple times. To protect the data privacy, Wang et al. [24] exploited the random masking technique to construct the first public data integrity auditing scheme supporting privacy preserving. Li et al. [25] proposed a data integrity auditing scheme which preserves data privacy from the TPA. Yu et al. [26] proposed a cloud storage auditing scheme with perfect data privacy preserving by making use of zero-knowledge proof. To relieve the user's computation burden of authenticator generation, Guan et al. [27] constructed a data integrity auditing scheme using indistinguishability obfuscation technique, which reduces the overhead for generating data authenticators. Li et al. [28] proposed a data integrity auditing scheme which contains a cloud storage server and a cloud audit server. In this scheme, the cloud audit server helps user to generate data authenticators before uploading data to the cloud storage server. Shen et al. [29] designed a light-weight data integrity auditing scheme, which introduced a Third Party Medium to generate authenticators and verify data integrity on behalf of users.

The data sharing is used widely in cloud storage scenarios. To protect the identity privacy of user, Wang et al. [30] proposed a shared data integrity auditing scheme based on the ring signature. Yang et al. [31] designed a remote data integrity auditing scheme for shared data, which supports both the identity privacy and the identity traceability. By using the homomorphic verifiable group signature, Fu et al. [32] proposed a privacy-aware remote data integrity auditing scheme for shared data. In order to achieve efficient user revocation, Wang et al. [33] designed a shared data integrity auditing scheme supporting user revocation by making use of the proxy re-signature. Based on the identity-based setting, Zhang et al. [34] constructed a cloud storage auditing scheme for shared data supporting real efficient user revocation. To realize the data sharing with sensitive information hiding, Shen et al. [35] designed an identity-based cloud storage auditing scheme for shared data.

Other aspects, such as eliminating certificate management [36] and key exposure resilience [9–11] in data integrity auditing have also been studied. However, all of existing remote data integrity auditing schemes do not take the problem of private key storage into account. In this paper, we explore how to achieve data integrity auditing scheme without private key storage for secure cloud storage.

B. *Organization*

The rest of this paper is organized as follows: We describe the system model, notations and the cryptographic knowledge in Section II. In Section III, we review some preliminaries. In Section IV and Section V, the detailed description of the modified BLS short signature and the proposed data integrity auditing scheme without private key storage are introduced respectively. In Section VI, the security analysis of the proposed data integrity auditing scheme is given. Finally, we evaluate the performance of our scheme in Section VII and conclude the paper in Section VIII.

II. System Model, Notations and Cryptographic Knowledge

A. *System model*

As illustrated in Fig. 1, the system model involves three types of entities: the user, the cloud, and the TPA. The cloud provides enormous data storage space to the user. The user has a large number of files to be uploaded to the cloud. The TPA is a public verifier who is delegated by the user to verify the integrity of the data stored in the cloud.

In the phase of user registration, the biometric data (e.g. fingerprint) is extracted from the user who wants to use the cloud storage service. When a data owner would like to upload data to the cloud, he firstly extracts biometric data as his fuzzy

TABLE I
NOTATIONS

Notation	Meaning
$\mathbf{w}$	An n-dimensional vector, which is denoted as $\mathbf{w}=\{w_1, w_2, ..., w_n\}$
W	An aggregated element, which is computed by $\prod_{i\in\{1,2,...n\}} w_i$
p	The smallest prime satisfying $W\,\vert\, p-1$
G_1, G_2	Multiplicative cyclic groups with prime order p
g, u	Two generators of group G_1
e	A bilinear pairing map $e: G_1 \times G_1 \rightarrow G_2$
Z_p^*	A prime field with nonzero elements
H	A cryptographic hash function, $H: \{0,1\}^* \rightarrow G_1$
z	An element of Z_p^* with order W
x, x'	The elements of Z_W
$\mathbf{y}, \mathbf{y}'$	Biometric data extracted from the user
$\mathbf{c}, \mathbf{c}'$	The sketches used to code and correct the error of biometric data
$m_i (i = 1, 2, ..., s)$	The i-th block of file F
$\sigma_i (i = 1, 2, ..., s)$	The authenticators of blocks $m_i(i = 1, 2, ..., s)$
$\Phi = \{\sigma_i\}_{1\leq i\leq s}$	The authenticator set of file F
α	The signature corresponding to file F
c	The number of challenged blocks
μ	The linear combination of data blocks
σ	An aggregated data authenticator

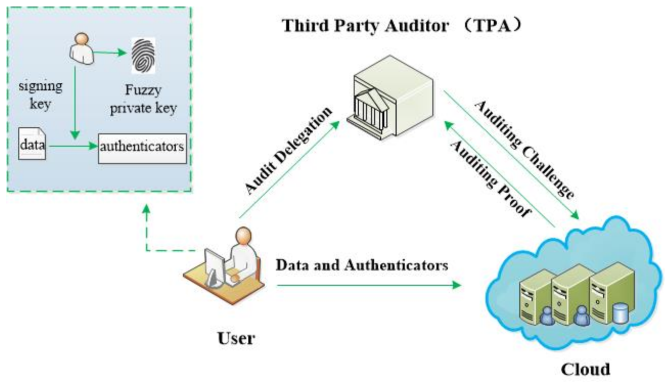

Fig. 1. System model of our data integrity auditing

private key and randomly generates a signing key. Then, this data owner computes authenticators for data blocks with his signing key. Finally, he uploads these data blocks along with the authenticator set to the cloud and deletes these messages from the local storage. In the phase of data integrity auditing, the TPA verifies whether the cloud truly keeps the user's intact data or not by executing the challenge-response protocol with the cloud.

B. Design Goals

To enable data integrity auditing without private key storage for secure cloud storage, our scheme should achieve the following goals:

1) *Auditing correctness*: to ensure that when the cloud properly stores users' data, the proof it generates can pass the verification of the TPA.
2) *Auditing soundness*: to assure that if the cloud does not possess users' intact data, it cannot pass the verification of the TPA.
3) *Auditing without private key storage*: to allow the user to utilize biometric data as fuzzy private key to accomplish data integrity auditing without private key storage.

C. Notations

In Table 1, we describe the notations used in the description of our scheme.

D. Cryptographic Knowledge

1) **Bilinear Maps**
Assume G_1 and G_2 are two multiplicative cyclic groups which have the same prime order p. A map $e: G_1 \times G_1 \rightarrow G_2$ is called bilinear map if it satisfies the following properties:
   a) *Bilinearity*: for all $u, v \in G_1$ and $a, b \in Z_p^*$, $e\left(u^a, v^b\right) = e(u,v)^{ab}$.
   b) *Non-degeneracy*: $e\left(g,g\right) \neq 1$, where g is a generator of G_1.
   c) *Computability*: there exists an efficiently computable algorithm for computing map $e: G_1 \times G_1 \rightarrow G_2$.

   Let *PPGen* be a bilinear groups generation algorithm (referred to as a bilinear group generator) which takes 1^k (k is a security parameter) as input, and generates a description of bilinear groups $PP = (p, G_1, G_2, g, e)$.
2) **Security Assumptions**
The security of our proposed scheme is based on the following security assumptions:
Computational Diffie-Hellman (CDH) Problem. Given g, g^x and $h \in G_1$, where $x \in Z_p^*$ is unknown, compute $h^x \in G_1$.
Definition 1: (Computational Diffie-Hellman (CDH) Assumption) The advantage of a probabilistic polynomial time algorithm $\mathcal{A}$ in solving the CDH problem in G_1 is

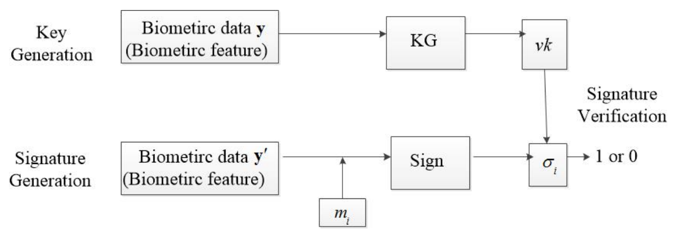

Fig. 2. The architecture of fuzzy signature

defined as

$$AdvCDH_{\mathcal{A}} = Pr[\mathcal{A}(g, g^x, h) = h^x : x \overset{R}{\leftarrow} Z_p^*, h \overset{R}{\leftarrow} G_1].$$

The probability is taken over the choice of x and h, and the coin tosses of $\mathcal{A}$. The CDH assumption means, for any polynomial time algorithm $\mathcal{A}$, the advantage that $\mathcal{A}$ solves the CDH problem in G_1 is negligible.

Discrete Logarithm (DL) Problem. Given g, $g^x \in G_1$, where $x \in Z_p^*$ are unknown, compute x.

Definition 2: (Discrete Logarithm (DL) Assumption) The advantage of a probabilistic polynomial time algorithm $\mathcal{A}$ in solving the DL problem in G_1 is defined as

$$AdvDL_{\mathcal{A}} = Pr[\mathcal{A}(g, g^x) = x : x \overset{R}{\leftarrow} Z_p^*].$$

The probability is taken over the choice of x, and the coin tosses of $\mathcal{A}$. The DL assumption means, for any polynomial time algorithm $\mathcal{A}$, the advantage that $\mathcal{A}$ solves the DL problem in G_1 is negligible.

III. Preliminaries

In this section, we provide some preliminaries of this paper, consisting of fuzzy signature, group isomorphism based on Chinese Remainder Theorem, coding and error correction and linear sketch.

A. Fuzzy Signature

The concept of fuzzy signature was proposed in [37]. It uses biometric data as private key, such as iris scan and fingerprint, to generate the signature. The biometric data $\mathbf{y}$ is a feature vector [38] which is defined as an *n*-dimensional vector in our scheme.

Fig. 2 presents the architecture of fuzzy signature. In a fuzzy signature scheme, the key generation algorithm *KeyGen* takes the biometric data $\mathbf{y}$ as input, and generates a verification key *vk*. The signature generation algorithm *SigGen* takes as input the biometric data $\mathbf{y}'$ and a data block m_i, and generates the signature σ_i of m_i. The verification algorithm *Verify* takes as input the verification key *vk*, the data block m_i and the signature σ_i, and verifies whether the signature σ_i is valid or not. If the biometric data $\mathbf{y}'$ is sufficiently close to the biometric data $\mathbf{y}$, it means that $\mathbf{y}'$ and $\mathbf{y}$ are extracted from the same user. Thus, the signature σ_i is valid; otherwise, it is invalid.

1 **Formalization of Fuzzy Key Setting**

In a typical biometric authentication scheme [39], biometric data $\mathbf{y} = (y_1, ..., y_n) \in \mathbf{Y}$ ($\mathbf{Y}$ is the metric space including all possible biometric data $\mathbf{y}$) is extracted from a user in the phase of registration. In the phase of authentication, biometric data $\mathbf{y}' = (y_1', ..., y_n') \in \mathbf{Y}$ is extracted from a user. If $\mathbf{y}'$ is sufficiently close to $\mathbf{y}$, we can conclude that the user who generated the biometric data $\mathbf{y}'$ and the user who generated the biometric data $\mathbf{y}$ are the same user; otherwise, they are different users.

A fuzzy key setting *FKS* includes $((d, \mathbf{Y}), \gamma, \varepsilon, \Omega, \theta)$ [37]. These symbols are defined as follows:

a) $(d, \mathbf{Y})$: This is a metric space, where $\mathbf{Y}$ ($\mathbf{Y} := [0, 1)^n \subset R^n$, *R* is the set of all real numbers) is the vector space including all possible biometric data $\mathbf{y} : (y_1, ..., y_n) \in \mathbf{Y}$, and $d : \mathbf{Y} \times \mathbf{Y} \rightarrow R^n$ is the corresponding distance function. We define $d(\mathbf{y}, \mathbf{y}') = \max_{i \in \{1,...,n\}} |y_i - y_i'|$ for vectors $\mathbf{y} = (y_1, ..., y_n), \mathbf{y}' = (y_1', ..., y_n') \in \mathbf{Y}$.
b) γ: This is a uniform distribution of biometric data over $\mathbf{Y}$.
c) ε: This is the threshold value which belongs to *R* and is determined by a security parameter *k* ($k = \lfloor -n\log_2(2\varepsilon) \rfloor$). We set that p_1 is the probability that two different users are accepted in the phase of identity authentication, it means that two different users are considered to be the same user. We require that this probability p_1 is negligible in *k* based on ε. In other words, if the distance between two different biometric data $\mathbf{y}$ and $\mathbf{y}'$ is less than ε (that is $d(\mathbf{y}, \mathbf{y}') < \varepsilon$), the probability p_1 is negligible in *k*.
d) Ω: This is an error distribution. If a user extracts biometric data $\mathbf{y}$ in the phase of registration, and extracts biometric data $\mathbf{y}'$ next time, $\mathbf{y}'$ follows the distribution $\{\mathbf{e} \leftarrow_R \Omega;\ \mathbf{y}' \leftarrow \mathbf{y} + \mathbf{e} : \mathbf{y}'\}$. We can know that $\mathbf{e}$ is the "noise" of the biometric data extracted from the same user and the error distribution Ω is independent of individual.
e) θ: This is an error parameter within $[0, 1]$. We can know that if $\mathbf{y}$ is the biometric data extracted from a user and $\mathbf{e} \leftarrow_R \Omega$ is the "noise" of the biometric data extracted from the same user, $\mathbf{y}+\mathbf{e}$ should be the biometric data extracted from the same user. We assume p_2 is the probability that the same user is rejected in the phase of identity authentication, it means that a user is considered to be two different users. We require that the probability p_2 is less than or equal to θ. That is, if $d(\mathbf{y}, \mathbf{y}+\mathbf{e}) \geq \varepsilon$, the probability $p_2 \leq \theta$.

2 **Definition of Fuzzy Signature**

A fuzzy signature scheme which is associated with a fuzzy key setting $FKS = ((d, \mathbf{Y}), \gamma, \varepsilon, \Omega, \theta)$ includes the following four algorithms:

a) $Setup$: The setup algorithm takes as input the description of the fuzzy key setting *FKS* and a security parameter *k* (*k* determines the threshold value ε of *FKS*), and generates a public parameter *pp*.
b) $KeyGen$: The key generation algorithm takes as input the public parameter *pp* and biometric data $\mathbf{y} \in \mathbf{Y}$, and generates a verification key *vk*.
c) $SigGen$: The signature generation algorithm takes as input a public parameter *pp*, biometric data $\mathbf{y}' \in \mathbf{Y}$ and a data block m_i, and generates the signature σ_i of m_i.

d) $Verify$: The verification algorithm takes as input a public parameter *pp*, a verification key *vk*, a data block m_i and the signature σ_i of m_i, and returns 1 or 0 to prove the signature σ_i is valid or not.

B. Group Isomorphism Based on Chinese Remainder Theorem

Let n be a natural number, and $w_1, \ldots, w_n \in N^+$ (N^+ is the set of all positive integers) be with the same bit length, such that

$$\forall i \in \{1, 2, \ldots, n\}, w_i \leq 1/(2\varepsilon), \tag{1}$$

and

$$\forall i \neq j \in \{1, 2, \ldots, n\}, gcd(w_i, w_j) = 1. \tag{2}$$

Set $W = \prod_{i \in \{1,2,\ldots,n\}} w_i = \Theta(2^k)$, where $k = \lfloor -n\log_2(2\varepsilon) \rfloor$.

We assume that P_1Gen is a deterministic algorithm. It takes ε and n as input, and generates a vector $\mathbf{w} = (w_1, ..., w_n)$ satisfying the above equations (1) and (2).

Set $\mathbf{v} \bmod \mathbf{w} = (v_1 \bmod w_1, ..., v_n \bmod w_n)$, where $\mathbf{v} = (v_1, ..., v_n) \in Z^n$ and $\mathbf{w} = (w_1, ..., w_n) \in Z^n$ (Z is the set of all integers). We define the equivalence relation "$\sim$" by $\mathbf{v}_1 \sim \mathbf{v}_2 \Leftrightarrow \mathbf{v}_1 \bmod \mathbf{w} = \mathbf{v}_2 \bmod \mathbf{w}$ for vectors $\mathbf{v}_1, \mathbf{v}_2 \in Z^n$. Let $Z^n_{\mathbf{w}} = Z^n/\sim$ be the quotient set of Z^n by $\sim$.

We randomly choose two vectors $\mathbf{v}, \mathbf{w} \in Z^n$, then find a value V satisfying $V \bmod w_i = v_i$ ($i \in \{1, 2, \ldots, n\}$). According to the Chinese Remainder Theorem (CRT), we know that the value V is determined uniquely modulo W. Therefore, we define a mapping $\mathrm{CRT}_{\mathbf{w}} : Z^n_{\mathbf{w}} \to Z_W$ for a fixed $\mathbf{w} = (w_1, ..., w_n) \in Z^n$. It means that $\mathrm{CRT}_{\mathbf{w}}(\mathbf{v}) = V \in Z_W$. Furthermore, $\mathrm{CRT}^{-1}_{\mathbf{w}}$ can be regarded as the "inverse" mapping of $\mathrm{CRT}_{\mathbf{w}}$, which is denoted by $Z_W \to Z^n_{\mathbf{w}}$.

Note that $\mathrm{CRT}_{\mathbf{w}}$ has the following homomorphism property: For any $\mathbf{v}_1, \mathbf{v}_2 \in Z^n_{\mathbf{w}}$, the equation $\mathrm{CRT}_{\mathbf{w}}(\mathbf{v}_1 + \mathbf{v}_2) = \mathrm{CRT}_{\mathbf{w}}(\mathbf{v}_1) + \mathrm{CRT}_{\mathbf{w}}(\mathbf{v}_2) \bmod W$ holds. $\mathrm{CRT}_{\mathbf{w}}$ is bijective between $Z^n_{\mathbf{w}}$ and Z_W, so $\mathrm{CRT}_{\mathbf{w}}$ is an isomorphism.

C. Coding and Error Correction

Let $\mathbf{w} = (w_1, ..., w_n) \in N^n$ (N is the set of all natural numbers) be the vector satisfying equations (1) and (2). Let $R^n_{\mathbf{w}} = R^n/\sim$ (R is the set of all real numbers) be the quotient set of R^n by $\sim$, which is similar to the definition of $Z^n_{\mathbf{w}}$. Let δ be the number satisfying $\delta = nw_i + r \in R$ (n is an integrity and $0 \leq r < w_i$).

Let $F_{\mathbf{w}} : R^n \to R^n_{\mathbf{w}}$ be the following function:

$$F_{\mathbf{w}}(\mathbf{y}) = (w_1y_1, ..., w_ny_n) \in R^n_{\mathbf{w}}. \tag{3}$$

We know that $F_{\mathbf{w}}(\mathbf{y}+\mathbf{e}) = F_{\mathbf{w}}(\mathbf{y}) + F_{\mathbf{w}}(\mathbf{e})(\mathrm{mod}\mathbf{w})$ holds. Thus, $F_{\mathbf{w}}$ can be regarded as a kind of linear coding.

Let $P_{\mathbf{w}} : R^n_{\mathbf{w}} \to Z^n_{\mathbf{w}}$ be the following function:

$$P_{\mathbf{w}}((\delta_1, ..., \delta_n)) = (\lfloor \delta_1 + 0.5 \rfloor, ..., \lfloor \delta_n + 0.5 \rfloor). \tag{4}$$

We know that if $\delta + 0.5 \in R$, then $\lfloor \delta + 0.5 \rfloor$ denotes the maximum integer that does not exceed $\delta + 0.5$ and can be viewed as the round-off operation. Therefore, $P_{\mathbf{w}}$ can be viewed as a kind of error correction. Specifically, according to equation (1), we know that $d(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) < \max_{i \in \{1,...,n\}}\{w_i\} \cdot \varepsilon \leq 0.5$ holds for any $\mathbf{y}, \mathbf{y}' \in Y$ and $d(\mathbf{y} - \mathbf{y}') < \varepsilon$.

Therefore, for such $\mathbf{y}$ and $\mathbf{y}'$, the following equation holds

$$P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) = 0. \tag{5}$$

Besides, for any $\mathbf{y} \in R^n$ and $\mathbf{x} \in Z^n_{\mathbf{w}}$, we have

$$P_{\mathbf{w}}(\mathbf{y} + \mathbf{x}) = P_{\mathbf{w}}(\mathbf{y}) + \mathbf{x}(\mathrm{mod}\mathbf{w}). \tag{6}$$

D. Linear sketch

Let $FKS = ((d, \mathbf{Y}), \gamma, \varepsilon, \Omega, \theta)$ be a fuzzy key setting defined previously. We design a linear sketch scheme which is used to code and correct the error. This scheme is similar to the one-time pad encryption scheme. In a one-time pad encryption scheme, a plaintext m's ciphertext c with a key sk is calculated as $c = m + sk$. The one-time pad encryption scheme satisfied the following property. For two ciphertexts $c = m + sk$ and $c' = m' + sk$ with the same key sk, the "difference" $\triangle m = m - m'$ of plaintexts can be computed by comparing c and c'. In the designed linear sketch scheme, we make use of the above one-time pad encryption's property. Thus, the process of coding in the linear sketch scheme can be viewed as the process of one-way encryption in the one-time pad encryption scheme, which is used to code the biometric data with a random value. And the process of correcting error in the linear sketch scheme can be viewed as the process of finding the "difference" in the one-time pad encryption scheme, which is used to compute the "difference" of two random values.

The description of linear sketch scheme is as follows:

Let $\mathbf{w}$ be a vector $(w_1, ..., w_n)$ generated by the threshold value ε of FKS and the dimension n of $\mathbf{Y}$. Set $W = \prod_{i \in \{1,2,...,n\}} w_i$. Let T be the description of an additive group $(Z_W, +)$.

1) **Coding**$(T, x, \mathbf{y})$:
   Given a random value $x \in Z_W$, biometric data $\mathbf{y} \in [0, 1)^n$ and the description T of additive group $(Z_W, +)$, this algorithm computes the sketch $\mathbf{c}$ as follows:

$$\mathbf{c} = (\mathrm{CRT}^{-1}_{\mathbf{w}}(x) + F_{\mathbf{w}}(\mathbf{y})) \bmod \mathbf{w}, \tag{7}$$

   where $\mathrm{CRT}^{-1}_{\mathbf{w}}$ is the "inverse" mapping of $\mathrm{CRT}_{\mathbf{w}}$ and $F_{\mathbf{w}}$ is the function of a kind of linear coding.
2) **ErrorCorrection**$(T, \mathbf{c}, \mathbf{c}')$:
   Given the description T, the sketch $\mathbf{c} = (\mathrm{CRT}^{-1}_{\mathbf{w}}(x) + F_{\mathbf{w}}(\mathbf{y})) \bmod \mathbf{w}$ and the sketch $\mathbf{c}' = (\mathrm{CRT}^{-1}_{\mathbf{w}}(x') + F_{\mathbf{w}}(\mathbf{y}')) \bmod \mathbf{w}$, this algorithm returns the difference between x and x' by computing $P_{\mathbf{w}}(\mathbf{c} - \mathbf{c}') = \Delta\mathbf{x}$ and $\mathrm{CRT}_{\mathbf{w}}(\Delta\mathbf{x}) = \Delta x$.
   Based on the properties of coding and error correction, the correctness of the above equations is presented as follows: $P_{\mathbf{w}}(\mathbf{c} - \mathbf{c}') = P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}') + (\mathrm{CRT}^{-1}_{\mathbf{w}}(x) - \mathrm{CRT}^{-1}_{\mathbf{w}}(x'))) = P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) + \mathrm{CRT}^{-1}_{\mathbf{w}}(x) - \mathrm{CRT}^{-1}_{\mathbf{w}}(x') = \mathrm{CRT}^{-1}_{\mathbf{w}}(x - x') = \Delta\mathbf{x}$ since $d(\mathbf{y} - \mathbf{y}') < \varepsilon$ then $P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) = 0$. In addition, $\mathrm{CRT}_{\mathbf{w}}(\Delta\mathbf{x}) = \mathrm{CRT}_{\mathbf{w}}(\mathrm{CRT}^{-1}_{\mathbf{w}}(x - x')) = x - x' = \Delta x$.

IV. The Modified BLS Short Signature (MBLSS)

To realize data integrity auditing without private key storage, how to design a special signature satisfying the compatibility with the linear sketch and the blockless verifiability is a key challenge. In the following, we firstly review the BLS short signature, and then describe the proposed signature scheme.

A. BLS Short Signature

Let G_1 and G_2 be two multiplicative cyclic groups with prime order p. Let g be a generator of G_1 and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. Let $H : \{0,1\}^* \rightarrow G_1$ be a cryptographic hash function. The global parameters are (G_1, G_2, p, g, e, H).

A BLS short signature scheme [40] consists of the following three algorithms: $KeyGen$, $SigGen$ and $Verify$. The description of this scheme is as follows:

1) **KeyGen**:
The user randomly selects $x \in Z_p^*$ as his private key, and generates his public key $\lambda = g^x \in G_1$.

2) **SigGen**:
Assume that file F is divided into s blocks $(m_1, m_2, ..., m_s)$. The user generates the signature σ_i of the block m_i with his private key x as follows: $\sigma_i = H(m_i)^x \in G_1$.

3) **Verify**:
The verifier checks the validity of the signature σ_i by verifying the following equation:

$$e(\sigma_i, g) = e(H(m_i), \lambda) \tag{8}$$

If the above equation holds, then this signature σ_i is correct; otherwise, it is wrong.

The correctness of the verification equation (8) can be shown as follows:

$$e(\sigma_i, g) = e(H(m_i)^x, g) = e(H(m_i), g^x) = e(H(m_i), \lambda).$$

B. The Proposed Modified BLS Short Signature

The original BLS short signature cannot be directly applied to remote data integrity auditing without private key storage, which faces the following challenges. Firstly, the original BLS short signature is incompatible with the linear sketch scheme because the signing key spaces of the two schemes are different. Secondly, the original BLS short signature does not support blockless verifiability, so the verifier has to download the entire data from the cloud to verify the integrity of data. It will incur excessive communication overhead and verification cost in big data scenario.

To address the above challenges, we design a modified BLS short signature named as MBLSS. The MBLSS is compatible with the linear sketch scheme, and allows a verifier to check the integrity of cloud data without downloading the whole data from the cloud. We will present how to achieve the remote data integrity auditing based on MBLSS in Section V.

1 **Description of the MBLSS**

In order to achieve the compatibility with the linear sketch and the blockless verifiability, we construct MBLSS by modifying the original BLS signature. The MBLSS is similar to the original BLS signature, except that:

a) Generate a different order for bilinear groups. In the original BLS signature, the order p of the bilinear groups G_1 and G_2 can be any large prime satisfying the basic security requirement. In contrast, the order p of the bilinear groups G_1 and G_2 in MBLSS must be selected as the smallest prime satisfying special condition so that the groups can be compatible with the linear sketch.
b) Select a signing key from a different space. The signing key is selected from Z_p^* in the original BLS signature, while it is selected from Z_W in MBLSS, where $W = \prod_{i \in \{1,2,...,n\}} w_i$ and w_i $(i \in \{1, 2, ..., n\})$ is an element of vector **w**.
c) Satisfy the following properties which homomorphic verifiable signature [19] meets:.
   - **Blockless verification**: Let σ_i and σ_j be the authenticators for data blocks $m_i, m_j \in Z_p^*$ respectively. Choose two random values r_i and r_j. The verifier is able to check the correctness of data block $m' = r_i m_i + r_j m_j \in Z_p^*$ without knowing blocks m_i and m_j.
   - **Non-malleability**: If an adversary does not have the private key of data owner, it cannot compute a valid authenticator σ' for block $m' = r_i m_i + r_j m_j \in Z_p^*$ by linearly combining σ_i and σ_j.

The details of the MBLSS scheme are as follows:

Let P_1Gen denote an algorithm, which takes as input ε and *n* (ε and *n* are the parameters of the fuzzy data setting *FKS* corresponding the security parameter *k*), and outputs a vector $\mathbf{w} = (w_1, ..., w_n) \in Z^n$ (Z is the set of all integers) satisfying the equations (1) and (2). Let $P_2Gen(W)$ denote an algorithm, which takes $W = \prod_{i \in \{1,2,...,n\}} w_i$ as input and outputs p which is the smallest prime satisfying $W|p-1$. Let PP_{MBLS} be another algorithm. It takes 1^k as input, runs the algorithms $P_1Gen(\varepsilon, n)$ and $P_2Gen(W)$, and outputs a description of bilinear groups $PP = (G_1, G_2, p, g, e)$, where G_1 and G_2 are multiplicative cyclic groups with prime order p, g is a generator of G_1 and $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map. Let *u* be the independent generator of G_1 and *H* be a cryptographic hash function $H : \{0,1\}^* \rightarrow G_1$. Let z be an element of Z_p^* with order *W*. The global parameters are $(G_1, G_2, p, g, e, u, H, z)$.

a) **KeyGen**:
The user randomly picks $x \in Z_W$ as his private key, and calculates $\lambda = g^{z^x}$ as his public key.

b) **AuthGen**:
Assume that a file F is divided into s blocks $(m_1, m_2, ..., m_s)$. The user computes a signing key $x' = z^x \bmod p$ which is used to generate the authenticator σ_i of the block m_i. The authenticator σ_i can be computed as $\sigma_i = (H(name||i) \cdot u^{m_i})^{x'}$, where $name \in Z_p^*$ is the identifier of the file *F*.

c) **Verify**:
The verifier checks the validity of the authenticator σ_i as follows :

$$e(\sigma_i, g) = e(H(name||i) \cdot u^{m_i}, \lambda) \tag{9}$$

If the above equation holds, then this authenticator σ_i is correct; otherwise, not.

2 **Security of the MBLSS**

We prove the MBLSS scheme is secure in terms of correctness, homomorphism and unforgeability.

Theorem 1: (Correctness) Given a data block m_i, its corresponding authenticator σ_i and the related public parameters, a verifier can verify the correctness of this data block.

Proof. Based on the properties of the bilinear map, the correctness of equation (9) is presented as follows:

$$\begin{aligned} e(\sigma_i, g) &= e((H(name\|i) \cdot u^{m_i})^{x'}, g) \\ &= e((H(name\|i) \cdot u^{m_i}, g^{x'}) \\ &= e((H(name\|i) \cdot u^{m_i}, g^{z^x}) \\ &= e((H(name\|i) \cdot u^{m_i}, \lambda). \end{aligned}$$

Therefore, the verifier can verify the correctness of data block by the above equation. As long as the block is intact, this equation will hold.

Theorem 2: (Homomorphism) The MBLSS is a homomorphic signature satisfying both blockless verifiability and non-malleability.

Proof. Given two data blocks $m_i, m_j \in Z_p^*$ and their corresponding authenticators σ_i and σ_j, we first prove that the MBLSS is able to achieve blockless verification, and then show that it is also non-malleable. Let r_i and r_j be two random values from Z_p^*.

Blockless Verifiability. A verifier is able to check the correctness of data block $m' = r_i m_i + r_j m_j \in Z_p^*$ without knowing blocks m_i and m_j by verifying the correctness of the following equation:

$$e({\sigma_i}^{r_i} {\sigma_j}^{r_j}, g) = e(H(name\|i)^{r_i} H(name\|j)^{r_j} \cdot u^{m'}, \lambda).$$

Based on Theorem 1, the correctness of the above equation can be proved as follows:

$$\begin{aligned} e(\sigma_i^{r_i}\sigma_j^{r_j}, g) &= e(((H(name\|i) \cdot u^{m_i})^{x'})^{r_i} \\ &\quad \cdot ((H(name\|j) \cdot u^{m_j})^{x'})^{r_j}, g) \\ &= e((H(name\|i) \cdot u^{m_i})^{r_i} \\ &\quad \cdot (H(name\|j) \cdot u^{m_j})^{r_j}, g^{x'}) \\ &= e((H(name\|i)^{r_i} \cdot u^{m_i r_i}) \\ &\quad \cdot (H(name\|j)^{r_j} \cdot u^{m_j r_j}), g^{x'}) \\ &= e(H(name\|i)^{r_i} H(name\|j)^{r_j} \cdot u^{m_i r_i + m_j r_j}, g^{z^x}) \\ &= e(H(name\|i)^{r_i} H(name\|j)^{r_j} \cdot u^{m'}, \lambda). \end{aligned}$$

If the above equation holds, the combined block m' is deemed to be correct and the verifier also believes that blocks m_i and m_j are both correct. Thus, the MBLSS can support blockless verification.

Non-malleability. An adversary, who does not possess the data owner's signing key, cannot generate the valid authenticator $\sigma' = {\sigma_i}^{r_i} \cdot {\sigma_j}^{r_j}$ for the combined block $m' = r_i m_i + r_j m_j \in Z_p^*$ by linearly combining σ_i and σ_j with r_i and r_j. The authenticator σ' cannot pass the verification equation (9) in the algorithm $Verify$:

$$\begin{aligned} e(\sigma', g) &= e({\sigma_i}^{r_i} \cdot {\sigma_j}^{r_j}, g) \\ &= e(((H(name\|i) \cdot u^{m_i})^{x'})^{r_i} \\ &\quad \cdot ((H(name\|j) \cdot u^{m_j})^{x'})^{r_j}, g) \\ &= e(H(name\|i)^{r_i} H(name\|j)^{r_j} \cdot u^{m'}, \lambda) \\ &\neq e(H(name') \cdot u^{m'}, \lambda). \end{aligned}$$

From the above equation, we can know that if the authenticator σ' of combined block m' can pass the verification equation (9), then $H(name') = H(name\|i)^{r_i} \cdot H(name\|j)^{r_j}$. Given a value $h = H(name\|i)^{r_i} \cdot H(name\|j)^{r_j}$, we can easily find a value $name'$ so that $H(name') = h$. This would contradict to the assumption that H is a one-way hash function. Therefore, the MBLSS is with non-malleability.

From the above analysis, we conclude that the MBLSS is a homomorphic signature meeting the property of blockless verifiability and non-malleability.

Theorem 3: (Unforgeability) For any adversary, it cannot generate a valid authenticator without the signing key of the data owner.

Proof. Assume that σ is a valid authenticator generated by the data owner, and σ' is a forged authenticator generated by the adversary. From the algorithm $AuthGen$ of the M-BLSS, we can know that a valid authenticator σ is generated with the data owner's signing key x'. If the adversary cannot obtain the signing key x' from the public key λ, then he cannot forge a valid authenticator σ. Therefore, the MBLSS is unforgeable if the DL problem in G_1 is computationally infeasible.

V. The Proposed Data Integrity Auditing Scheme without Private Key Storage

A. Definition of Data Integrity Auditing Scheme without Private Key Storage

A data integrity auditing scheme without private key storage consists of the following five algorithms: *Setup*, *KeyGen*, *SignGen*, *ProofGen* and *ProofVerify*. Specifically, these algorithms are described as follows:

1) *Setup*$(1^k, FKS)$: This algorithm takes as input a fuzzy key setting *FKS* and a security parameter *k*. It outputs the public parameter *pp*′.
2) $KeyGen(pp', \mathbf{y})$: This algorithm takes as input the public parameter *pp*′ and the biometric data $\mathbf{y} \in R^n$. It generates *pk* as his public key, which including a sketch **c** and a verification key *vk*.
3) $SignGen(\mathbf{y}', F)$ This algorithm takes as input the biometric data $\mathbf{y}' \in R^n$ and the file *F*. It outputs a signature α which includes the verification key vk', the sketch $\mathbf{c}'$ and the set of authenticators Φ.
4) $ProofGen(F, \Phi, chal)$ This algorithm takes as input the file *F*, the corresponding authenticator set Φ and the auditing challenge *chal*. It outputs an auditing proof *P* that proves the cloud indeed keeps this file.
5) $ProofVerify(pk, chal, P, vk', \mathbf{c}')$ This algorithm takes as input the user's public key pk, the auditing challenge

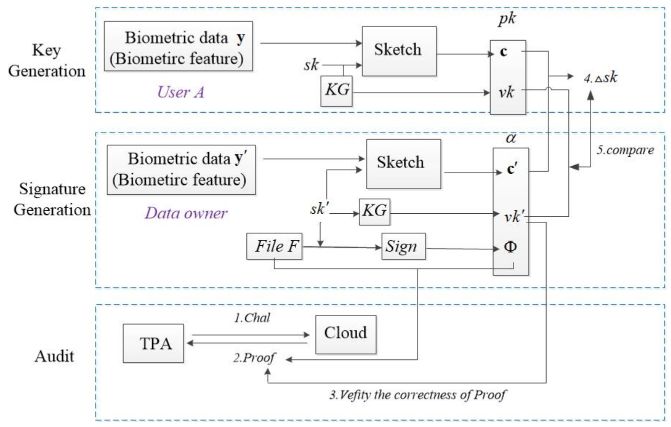

Fig. 3. An overview of data integrity auditing scheme without private key storage

chal, the auditing proof *P*, the verification key vk' and the sketch $\mathbf{c}'$. The TPA verifies the correctness of proof *P*.

B. Overview of the Proposed Scheme

Our proposed data integrity auditing scheme without private key storage is constructed based on the MBLSS and the linear sketch. As shown in Fig. 3, our proposed data integrity auditing scheme consists of the following three procedures: Key Generation, Signature Generation and Audit.

Key Generation. It includes $Setup$ and $KeyGen$ algorithms. Firstly, the public global parameter pp' is generated in $Setup$ algorithm. In the $KeyGen$ algorithm, the user *A*, who wants to store his data in the cloud, extracts biometric data $\mathbf{y}$ in the phase of registration. Next, this user randomly generates a key pair (sk, vk). Finally, this user generates a sketch $\mathbf{c}$ of private key sk using $\mathbf{y}$, which is used to code and correct the error of biometric data. The public key *pk* of our proposed scheme includes $(vk, \mathbf{c})$.

Signature Generation. It consists of the $SignGen$ algorithm. The data owner generates the signature of the file *F*, and uploads this file along with its signature to the cloud. Specifically, the data owner randomly generates a signing key sk' and its corresponding verification key vk', where sk' is used to generate the sketch and the authenticators. Then the data owner generates a sketch $\mathbf{c}'$ of signing key sk' using the biometric data $\mathbf{y}'$ extracted from him. He generates a data authenticator set Φ for file *F* with signing key sk'. The signature α of file *F* is $(\Phi, vk', \mathbf{c}')$. The data owner sends $\{F, \alpha\}$ to the cloud, and deletes them from the local storage.

Audit. The $ProofGen$ algorithm and $ProofVerify$ algorithm are executed in this phase. In the $ProofGen$ algorithm, the TPA sends an auditing challenge *chal* to the cloud. Upon receiving the *chal*, the cloud returns an auditing proof *P* to the TPA. In the $ProofVerify$ algorithm, the TPA firstly checks the correctness of the proof *P* using the verification key vk'. And then, in order to confirm the identity of the data owner, the TPA recovers Δsk from $\mathbf{c}$ and $\mathbf{c}'$ by using the technique of coding and error correction. Finally, the TPA verifies whether the difference between vk and vk' truly corresponds to Δsk. If it does, the data owner is the user *A*; otherwise, he is not.

C. Description of the Proposed Scheme

1) **Setup**$(1^k, FKS)$

   The bilinear groups $PP = (G_1, G_2, p, g, e)$ of our proposed scheme are the same as that of the MBLSS scheme, where *p* is the smallest prime satisfying $W|p-1$, G_1 and G_2 are multiplicative cyclic groups with the prime order *p*, *g* is a generator of G_1 and $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map. Let *u* be the independent generator of G_1 and *H* be a cryptographic hash function $H : \{0,1\}^* \rightarrow G_1$. Let z be an element of Z_p^* with order *W*. The global parameters are $(G_1, G_2, p, g, e, u, H, z)$.

2) **KeyGen**$(pp', \mathbf{y})$

   In the phase of registration, biometric data $\mathbf{y} = (y_1, ..., y_n) \in R^n$ is extracted from the user *A* who wants to use the cloud storage service. The user *A* randomly selects $x \in Z_W$ as his private key *sk*, and generates $\lambda = g^{z^x}$ as his verification key *vk*. Next, the user *A* generates a sketch $\mathbf{c}$ used to code and correct the error of biometric data as follows:

$$\mathbf{c} = (\mathrm{CRT}_{\mathbf{w}}^{-1}(x) + F_{\mathbf{w}}(\mathbf{y})) \bmod \mathbf{w}. \tag{10}$$

   Publish $(vk, \mathbf{c})$ as the public key *pk*.

3) **SignGen**$(\mathbf{y}', F)$

   The data owner divides his outsourcing file *F* into *s* blocks which is denoted as $F = (m_1, ..., m_s)$.

   In the phase of signature generation, biometric data $\mathbf{y}' = (y_1', ..., y_n') \in R^n$ is extracted from the data owner. The data owner randomly chooses $x' \in Z_W$ as his signing key sk', and calculates $\lambda' = g^{z^{x'}}$ as his verification key vk'. The data owner computes the sketch $\mathbf{c}'$ as follows:

$$\mathbf{c}' = (\mathrm{CRT}_{\mathbf{w}}^{-1}(x') + F_{\mathbf{w}}(\mathbf{y}')) \bmod \mathbf{w}. \tag{11}$$

   Next, the data owner generates the authenticator σ_i of the data block m_i with his signing key x' as follows:

$$\sigma_i = \left(H\left(name||i\right) \cdot u^{m_i}\right)^{z^{x'}}, \tag{12}$$

   where $name \in Z_p^*$ is the identifier of the file *F*. Let $\Phi = \{\sigma_i\}_{1 \le i \le s}$ be a set of authenticators. We define $(\Phi, vk', \mathbf{c}')$ as the signature α.

   Finally, the data owner sends $\{F, \alpha\}$ to the cloud, and deletes the file *F* and its corresponding signature from the local storage.

4) **ProofGen**$(F, \Phi, chal)$
   a) To verify the integrity of cloud data, the TPA randomly selects a *c*-element subset *I* of set $[1, s]$, and generates a random $\beta_i \in Z_p^*$ for each $i \in I$. The TPA sends the auditing challenge $chal = \{i, \beta_i\}_{i \in I}$ to the cloud.
   b) Upon receiving the auditing challenge, the cloud computes a linear combination of data blocks $\mu = \sum_{i \in I} m_i \beta_i$ and an aggregated data authenticator $\sigma = \prod_{i \in I} {\sigma_i}^{\beta_i}$. Then, the cloud returns an auditing proof $P = \{\mu, \sigma, vk', \mathbf{c}'\}$ to the TPA.

5) **ProofVerify**$(pk, chal, P, vk', \mathbf{c}')$

The TPA checks whether the following equation holds.

$$e(\sigma, g) = e\left(\prod_{i \in I} H(name||i)^{\beta_i} \cdot u^{\mu}, vk'\right) \quad (13)$$

If the equation (13) does not hold, then the data stored in the cloud is corrupted; otherwise, the TPA does as follows:

The TPA recovers Δx from $\mathbf{c}$ and $\mathbf{c}'$ by computing $P_{\mathbf{w}}(\mathbf{c} - \mathbf{c}') = \Delta \mathbf{x}$ and $\mathrm{CRT}_{\mathbf{w}}(\Delta \mathbf{x}) = \Delta x$, which can confirm the identity of the user.

Then, he verifies the following equation holds or not:

$$(vk)^{z^{\Delta x}} = vk' \quad (14)$$

If the equation (14) holds, then the data stored in the cloud is intact; otherwise, it is corrupted.

VI. Security Analysis

In this section, we prove that the proposed scheme is secure in terms of the auditing correctness, the soundness of linear sketch and the auditing soundness.

Theorem 4: (Auditing Correctness) When the cloud properly stores data, the proof it generates can pass the TPA's verification.

Proof. Given a valid proof $P = \{\mu, \sigma, vk', \mathbf{c}'\}$ from the cloud, the verification equation (13) in algorithm $ProofVerify$ will hold. Based on the properties of bilinear maps and Theorem 1, the correctness of equation (13) is presented as follows:

$$\begin{aligned} e(\sigma, g) &= e(\prod_{i \in I} {\sigma_i}^{\beta_i}, g) \\ &= e((\prod_{i \in I} (H(name\|i) \cdot u^{m_i})^{z^{x'}})^{\beta_i}, g) \\ &= e(\prod_{i \in I} (H(name\|i) \cdot u^{m_i})^{\beta_i}, g^{z^{x'}}) \\ &= e(\prod_{i \in I} (H(name\|i)^{\beta_i} \cdot u^{\sum_{i \in I} m_i \beta_i}, vk') \\ &= e(\prod_{i \in I} (H(name\|i)^{\beta_i} \cdot u^{\mu}, vk') \end{aligned}$$

Therefore, if the cloud properly stores the user's data, it will pass the TPA's verification.

Theorem 5: (the Soundness of Linear Sketch) The data owner can pass the verification of the TPA if the biometric data $\mathbf{y}$ and $\mathbf{y}'$ are both extracted from him; otherwise, he cannot pass the TPA's verification.

Proof. Set $x' = x + \Delta x$, where $x, x', \Delta x \in Z_W$. Let $vk = g^{z^x}$ and $vk' = g^{z^{x'}}$. Assume that biometric data $\mathbf{y}$ and $\mathbf{y}'$ are both extracted from the data owner. Let s-ketch $\mathbf{c} = (\mathrm{CRT}_{\mathbf{w}}^{-1}(x) + F_{\mathbf{w}}(\mathbf{y})) \bmod \mathbf{w}$ and sketch $\mathbf{c}' = (\mathrm{CRT}_{\mathbf{w}}^{-1}(x') + F_{\mathbf{w}}(\mathbf{y}')) \bmod \mathbf{w}$. According to coding and error correction, we can know that if $\mathbf{y}, \mathbf{y}' \in \mathbf{Y}$ are the biometric data satisfying $d(\mathbf{y} - \mathbf{y}') < \varepsilon$, the following two equations $P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) = \mathbf{0}$ and $P_{\mathbf{w}}(\mathbf{y} + \mathbf{x}) = P_{\mathbf{w}}(\mathbf{y}) + \mathbf{x}(\mathrm{mod}\mathbf{w})(\mathbf{x} \in Z_{\mathbf{w}}^n)$ hold. Based on the properties of coding and error correction and the linear sketch, we have

$$\begin{aligned} P_{\mathbf{w}}(\mathbf{c} - \mathbf{c}') =& P_{\mathbf{w}}(\mathrm{CRT}_{\mathbf{w}}^{-1}(x) + F_{\mathbf{w}}(\mathbf{y}) - (\mathrm{CRT}_{\mathbf{w}}^{-1}(x') + \\ & F_{\mathbf{w}}(\mathbf{y}'))) \\ =& P_{\mathbf{w}}(\mathrm{CRT}_{\mathbf{w}}^{-1}(x) - \mathrm{CRT}_{\mathbf{w}}^{-1}(x') + F_{\mathbf{w}}(\mathbf{y}) - \\ & F_{\mathbf{w}}(\mathbf{y}')) \\ =& P_{\mathbf{w}}(\mathrm{CRT}_{\mathbf{w}}^{-1}(x - x') + F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) \\ =& \mathrm{CRT}_{\mathbf{w}}^{-1}(x - x') + P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) \\ =& \mathrm{CRT}_{\mathbf{w}}^{-1}(x - x') \\ =& \Delta \mathbf{x}. \end{aligned}$$

Thus, $\mathrm{CRT}_{\mathbf{w}}(\Delta \mathbf{x}) = \Delta x$ and $(vk)^{z^{\Delta x}} = (g^{z^x})^{z^{\Delta x}} = g^{z^{x + \Delta x}} = g^{z^{x'}} = vk'$. It means that the data owner can pass the verification of the TPA.

If $\mathbf{y}, \mathbf{y}' \in \mathbf{Y}$ are the biometric data extracted from different users, which means that $d(\mathbf{y} - \mathbf{y}') > \varepsilon$ and $P_{\mathbf{w}}(F_{\mathbf{w}}(\mathbf{y}) - F_{\mathbf{w}}(\mathbf{y}')) \neq \mathbf{0}$, we have $P_{\mathbf{w}}(\mathbf{c} - \mathbf{c}') \neq \Delta \mathbf{x}$ and $(vk)^{z^{\Delta x}} \neq vk'$. Thus, the data owner cannot pass the TPA's verification.

Hence, if the biometric data $\mathbf{y}$ and $\mathbf{y}'$ are both extracted from the data owner, the data owner can pass the verification of the TPA; otherwise, he cannot pass the TPA's verification.

Theorem 6: (Auditing Soundness) If the CDH assumption holds in G_1 and the signature scheme used for generating file is existentially unforgeable, then except with negligible probability, an untrusted cloud can never pass verification of the TPA unless the cloud data has been preserved intact.

Proof. We will prove this theorem by employing the method of knowledge proof [21]. If the cloud can pass the TPA's verification without possessing the intact data, then we can extract the intact challenged data blocks by repeatedly interacting between the knowledge extractor and the scheme. We will accomplish our proof by a sequence of games.

Game 1. The challenger generates a signing key sk' used to generate authenticators and its corresponding verification key vk'. The adversary can get the authenticators of a series of data blocks by querying the challenger. Specifically, when the adversary wants to query data block m_i, he will submit a query to the challenger. The challenger calculates and sends the corresponding authenticator σ_i to the adversary. Then the challenger generates and sends an auditing challenge $chal = \{i, \beta_i\}_{i \in I}$ to the adversary. Upon receiving the challenge, the adversary responds the corresponding proof $P = \{\mu, \sigma, vk', \mathbf{c}'\}$ to the challenger, where $\mathbf{c}'$ is a sketch.

Game 2. Game 2 is the same as Game 1, with one difference. That is the challenger maintains a list of responses to the queries from the adversary. The challenger observes each instance of the challenge and respond process with the adversary. The challenger declares failure and aborts if he finds the aggregated authenticator σ is not equal to $\prod_{i \in I} {\sigma_i}^{\beta_i}$.

Analysis. Assume that an honest prover generates a correct proof $\{\mu, \sigma, vk', \mathbf{c}'\}$. From the correctness of our scheme, we have:

$$e(\sigma, g) = e\left(\prod_{i \in I} H(name\|i)^{\beta_i} \cdot u^{\mu}, vk'\right) \quad (15)$$

Assume that the adversary generates a proof $\{\mu', \sigma', vk', \mathbf{c}'\}$ which is different from the honest prover generated. Because the forgery is successful, we have:

$$e(\sigma', g) = e\left(\prod_{i\in I} H(name\|i)^{\beta_i} \cdot u^{\mu'}, vk'\right) \quad (16)$$

Obviously, $\mu \neq \mu'$, otherwise $\sigma = \sigma'$, which contradicts our assumption that the challenger declares failure and aborts. We define $\Delta\mu = \mu - \mu'$, and construct a simulator that could solve the challenge CDH instance with this adversary as follows:

Given $(g, g^{\upsilon}, h) \in G_1$, where $\upsilon = z^{x'}$, the simulator outputs h^{υ}. Set $u = g^a h^b$, where $a, b \in Z_p^*$ are random values chosen by the simulator. Meanwhile, the user's verification key is set as $vk' = g^{z^{x'}} = g^{\upsilon}$.

The simulator randomly selects $r_i \in Z_p^*$ for each $i (1 \le i \le s)$ in the challenge, and programs the random oracle at i as

$$H(name\|i) = g^{r_i}/(g^{am_i} \cdot h^{bm_i}) \quad (17)$$

The simulator can calculate σ_i, since we have

$$\begin{aligned} H(name\|i) \cdot u^{m_i} &= g^{r_i}/(g^{am_i} \cdot h^{bm_i}) \cdot u^{m_i} \\ &= g^{r_i}/(g^{am_i} \cdot h^{bm_i}) \cdot (g^a h^b)^{m_i} \\ &= g^{r_i}/(g^{am_i} \cdot h^{bm_i}) \cdot (g^{am_i} \cdot h^{bm_i}) \\ &= g^{r_i} \end{aligned}$$

Hence, the simulator computes $\sigma_i = (H(name\|i) \cdot u^{m_{\text{i}}})^{z^{x'}} = (g^{r_i})^{z^{x'}} = (g^{z^{x'}})^{r_i}$.

When dividing equation (16) by equation (15), we can get

$$\begin{aligned} e(\sigma'/\sigma, g) &= e(\prod_{i=1}^{s} ({\sigma'}_i/\sigma_i)^{\beta_i}, g) \\ &= e(\prod_{i=1}^{s} (u^{m_i'\beta_i}/u^{m_i\beta_i})^{z^{x'}}, g) \\ &= e(u^{\sum_{i=1}^{s} m_i'\beta_i - m_i\beta_i}, g^{z^{x'}}) \\ &= e(u^{\Delta\mu}, vk') \\ &= e((g^a h^b)^{\Delta\mu}, vk'). \end{aligned}$$

So, we know that

$$e(\sigma' \cdot \sigma^{-1} \cdot (vk')^{-a\Delta\mu}, g) = e(h, vk')^{b\Delta\mu}. \quad (18)$$

From the equation (18) and $vk' = g^{\upsilon}$, we can know that $h^{\upsilon} = (\sigma' \cdot \sigma^{-1} \cdot (vk')^{-a\cdot\Delta\mu})^{1/(b\cdot\Delta\mu)}$. Note that the probability of game failure is the same as the probability of $b \cdot \Delta\mu = 0 \bmod p$. The probability of $b \cdot \Delta\mu = 0 \bmod p$ is $1/p$ which is negligible, because p is a large prime. Then, we can find a solution to the CDH problem with a probability of $1 - 1/p$. It contradicts the assumption that the CDH problem in G_1 is computationally infeasible.

It means that if there is a non-negligible difference between the adversary's probabilities of success in Game 1 and Game 2, the constructed simulator can utilize the adversary to solve the CDH problem.

Game 3. Game 3 is the same as Game 2, with one difference. The challenger still keeps and observes each instance of the proposed auditing scheme. For one of these instances, the challenger declares failure and aborts if there exists a linear combination μ' of data blocks not equal to the expected μ.

Analysis. Assume that an honest prover generates a correct proof $\{\mu, \sigma, vk', \mathbf{c}'\}$. From the correctness of the scheme, we have $e(\sigma, g) = e\left(\prod_{i\in I} H(name\|i) \cdot u^{\mu}, vk'\right)$. Assume that the adversary generates a proof $\{\mu', \sigma', vk', \mathbf{c}'\}$ which is different from that the honest prover generated. Because the forgery is successful, we also have $e(\sigma', g) = e(\prod_{i\in I} H(name\|i)^{\beta_i} \cdot u^{\mu'}, vk')$. Based on the Game 2, we know that $\sigma' = \sigma$. Define $\Delta\mu = \mu' - \mu$. We construct a simulator that could solve the challenge DL instance with this adversary. The detailed process is as follows:

Given $(g, h) \in G_1$, the goal of the simulator is to calculate a value x which satisfies $h = g^x$. Set $u = g^a h^b$, where $a, b \in Z_p^*$ are random values chosen by the simulator.

From the analysis of the above two verification equations, we obtain

$$\begin{aligned} & e(\prod_{i\in I} H(name\|i)^{\beta_i} \cdot u^{\mu}, vk') \\ &= e(\sigma, g) \\ &= e(\sigma', g) \\ &= e(\prod_{i\in I} H(name\|i)^{\beta_i} \cdot u^{\mu'}, vk') \end{aligned}$$

Thus, we know that

$$u^{\mu} = u^{\mu'},$$

and therefore that

$$1 = u^{\Delta\mu} = (g^a h^b)^{\Delta\mu} = g^{a\Delta\mu} \cdot h^{b\Delta\mu}.$$

Then the solution to the DL problem is,

$$h = g^{-\frac{a\Delta\mu}{b\Delta\mu}} = g^{-\frac{a}{b}}.$$

However, b is zero only with the probability $1/p$, which is negligible because p is a large prime. Then, we can find a solution to the DL problem with a probability of $1 - 1/p$, which contradicts the assumption that the DL problem in G_1 is computationally infeasible.

It means that if there exists a non-negligible difference between the adversary's probabilities of success in Game 2 and Game 3, the constructed simulator can utilize the adversary to solve the DL problem.

Note that the hardness of the CDH problem implies the hardness of the discrete logarithm problem. Therefore, the differences between these games defined can be ignored if the CDH assumption in G_1 holds.

Finally, we construct a knowledge extractor to extract all of the challenged data blocks $m_i (i \in I, |i| = c)$. By selecting c different coefficients $\beta_i (i \in I, |I| = c)$ to execute c times different challenges on the same data blocks $m_i (i \in I, |I| = c)$. The knowledge extractor can get $m_i (i \in I, |I| = c)$ independently linear equations in the variables $m_i (i \in I, |I| = c)$, and then it can extract $m_i (i \in I, |I| = c)$ by solving these equations. It means that if the cloud does not possess users' intact data, it cannot pass the verification of the TPA.

Theorem 7: (The detectability): Our data integrity auditing scheme for secure cloud storage is $\left(\frac{k}{s}, 1 - \left(\frac{s-k}{s}\right)^c\right)$ detectable if the cloud stores a file F with s data blocks including k modified or deleted data blocks, and c data blocks are challenged.

Proof. Let the modified or deleted block-signature pairs be *Y*. Let the challenged block-signature pairs be *S*. Let the

TABLE II
COMPARISON ON COMPUTATION OVERHEAD OF EACH ENTITY IN OUR SCHEME AND SCHEME [21]

Scheme	Cloud	TPA	User
Our scheme	$(c-1)Mul_{G_1} + cExp_{G_1} + cMul_{Z_p^*} + (c-1)Add_{Z_p^*}$	$(c+2)Exp_{G_1} + cMul_{G_1} + 2Pair + cHash_{G_1} + Exp_{Z_p^*} + P + C$	$s(Hash_{G_1} + Mul_{G_1}) + 2sExp_{G_1} + sExp_{Z_p^*}$
Scheme [21]	$(c-1)Mul_{G_1} + cExp_{G_1} + cMul_{Z_p^*} + (c-1)Add_{Z_p^*}$	$(c+1)Exp_{G_1} + cMul_{G_1} + 2Pair + cHash_{G_1}$	$s(Hash_{G_1} + Mul_{G_1}) + 2sExp_{G_1}$

random variable set be $X = |Y \cap S|$. We use P_X to denote the probability of detecting the modified or deleted data blocks. That is to say, if the cloud modifies or deletes k data blocks of the file F, the cloud's misbehavior can be detected with probability P_X by challenging c data blocks. Thus, we get

$$\begin{aligned} P_X =& P\{X \geq 1\} \\ =& 1 - P\{X = 0\} \\ =& 1 - \frac{s-k}{s} \times \frac{s-1-k}{s-1} \times ... \times \frac{s-c+1-k}{s-c+1} \end{aligned}$$

We can have that $1 - (\frac{s-k}{s})^c \leq P_X \leq 1 - (\frac{s-c+1-k}{s-c+1})^c$. Therefore, we can conclude that the proposed scheme can detect the misbehavior of the cloud with probability at least $1 - \left(\frac{s-k}{s}\right)^c$ if the cloud stores a file F with s data blocks including k modified or deleted data blocks, and c data blocks are challenged.

VII. PERFORMANCE EVALUATION

In this section, we first analyze the computation complexity of our scheme, and then give the comparison between our scheme and Shacham and Waters's scheme [21] in terms of computation overhead and commutation overhead. Finally, we evaluate the performance of our scheme in experiments.

A. Performance Analysis and Comparison

We define the following notations to denote the operations in our scheme: s is the total number of data blocks. c is the number of challenged blocks. $Hash_{G_1}$, Exp_{G_1} and Mul_{G_1} respectively denote one hashing operation, one exponentiation operation and one multiplication operation in G_1. $Exp_{Z_p^*}$, $Mul_{Z_p^*}$ and $Add_{Z_p^*}$ respectively denote one exponentiation operation, one multiplication operation and one addition operation in Z_p^*. $Pair$ denotes one pairing operation, P denotes one $P_{\mathbf{w}}$ function operation and C denotes one $\text{CRT}_{\mathbf{w}}$ mapping operation. $|s|$ is the size of an element of set $[1, s]$, $|p|$ is the size of an element of Z_p^*, $|q|$ is the size of an element of G_1, and $|W|$ is the size of an element of Z_W.

TABLE III
COMPARISON ON COMMUNICATION OVERHEAD IN OUR SCHEME AND SCHEME [21]

Scheme	Communication overhead on the TPA side	Communication overhead on the cloud side
Ours	$c \cdot (\lvert s\rvert + \lvert p\rvert)$	$\lvert p\rvert + 2\lvert q\rvert + \lvert W\rvert$
Scheme [21]	$c \cdot (\lvert s\rvert + \lvert p\rvert)$	$\lvert p\rvert + \lvert q\rvert$

TABLE IV
THE COMPUTATION COMPLEXITY OF DIFFERENT ENTITIES IN DIFFERENT PHASES

	User	TPA	Cloud
Authenticator generation	$O(s)$	—	—
Challenge generation	—	$O(c)$	—
Proof generation	—	—	$O(c)$
Proof verification	—	$O(c)$	—

(1) **Comparison.** To evaluate the efficiency of our scheme, we choose scheme [21] as a benchmark. The construction of the scheme [21] is generally viewed as the most efficient one among all existing data integrity auditing schemes. However, our scheme can realize data integrity auditing without private key storage that is impossible to be achieved in scheme [21]. By the following comparison, we can conclude that our scheme only adds reasonable overhead to realize data integrity auditing without private key storage compared with the scheme [21].

Table II gives the computation overhead's comparison between our scheme and Shacham and Waters' scheme [21]. In the scheme [21], to reduce the storage overhead, one data block is divided into multiple sectors. Actually, our scheme also can support multiple sectors. But for simplification, we only consider that the file F is divided into s data blocks, and do not consider that each data block is divided into l sectors. For fairness comparison, we set $l = 1$ in the scheme [21]. As shown in Table II, on the user side, we can see that the computation overhead of calculating data authenticators in the scheme [21] is $s(Hash_{G_1} + Mul_{G_1}) + 2sExp_{G_1}$. Our scheme costs $s(Hash_{G_1} + Mul_{G_1}) + 2sExp_{G_1} + sExp_{Z_p^*}$ to generate data authenticators, which has nearly the same efficiency level with the scheme [21] on the user side. In the phase of proof generation, both our scheme and the scheme [21] cost the same computation overhead on the cloud side. In the phase of proof verification, our scheme only adds one Exp_{G_1}, one $Exp_{Z_p^*}$, one P and one C compared with the scheme [21].

Table III presents the communication overhead's comparison between our scheme and Shacham and Waters' scheme [21]. From Section V, we can know the communication overhead of our scheme is mainly from the interaction between the TPA and the cloud, which consists of auditing challenge and auditing proof. The size of an auditing challenge $chal = \{i, \beta_i\}_{i \in I}$ is $c \cdot (|s| + |p|)$ bits. The size of an auditing proof $P = \{\mu, \sigma, vk', \mathbf{c}'\}$ is $|p| + 2|q| + |W|$ bits. Therefore, in our scheme, the communication overhead of one auditing task

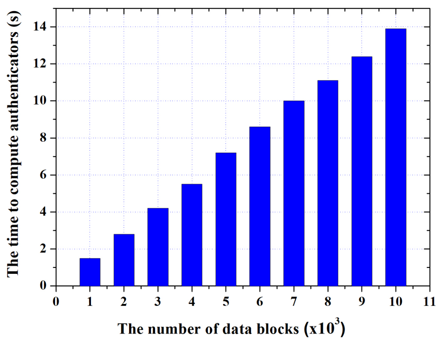

Fig. 4. The computation overhead of authenticator generation

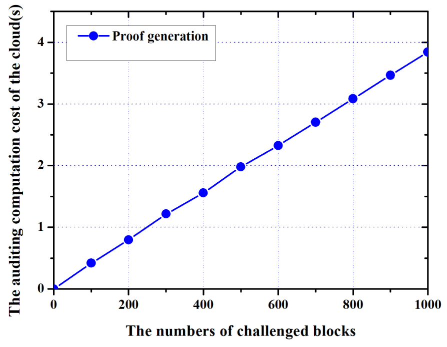

Fig. 6. The computation overhead of the cloud in the phase of auditing

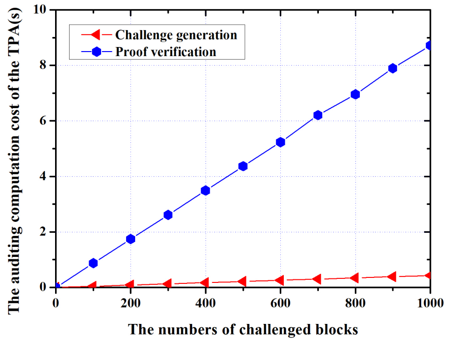

Fig. 5. The computation overhead of the TPA in the phase of auditing

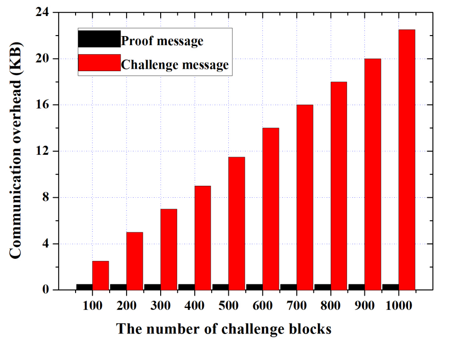

Fig. 7. The communication overhead of challenge message and proof message

is $c \cdot |s| + (c+1) \cdot |p| + 2|q| + |W|$ bits. In addition, comparing with the scheme [21], our scheme only incurs extra $|q| + |W|$ bits when executing one auditing task.

(2) **Computation complexity.** In Table IV, we analyze the computation complexity of the different entities in different phases. The computation complexity is about the number s of data blocks and the number c of challenged blocks. As shown in Table IV, the computation complexity of data authenticator generation on the user side is $O(s)$. On the TPA side, the computation complexity of challenge generation and proof verification are both $O(c)$. The computation complexity of proof generation is $O(c)$ on the cloud side.

B. Experimental Results

In this section, we evaluate the performance of our proposed scheme in experiments. We run these experiments on a Linux machine with an Intel Pentium 2.70GHz processor and 4GB memory. Our scheme is implemented by utilizing C programming language with the GNU Multiple Precision Arithmetic (GMP) Library [41] and the free Pairing-Based Cryptography (PBC) Library [42]. We set the base field size to be 512 bits, the size of an element in Z_p^* to be 160 bits and the size of a shared cloud file to be 20MB.

1) **Computation overhead**

a) **Authenticator generation.** In order to evaluate the efficiency of authentication generation of our scheme, we compute the authenticators for different blocks from 0 to 1000 increased by an interval of 100. Fig. 4 shows that the computation overhead of authenticator generation linearly increases with the number of data blocks. The running time varies from 1.5s to 12.9s.

b) **Auditing.** In order to evaluate the performance of auditing in our scheme, we respectively show the time spent on the TPA and the cloud. The experimental results are presented in Fig. 5 and Fig. 6. In the experiment, we choose to challenge different blocks from 0 to 1000 increased by an interval of 100. From Fig. 5, we have the observation that the auditing computation overhead of the TPA is mainly from challenge generation and proof verification. The running time of challenge generation ranges from 0.038s to 0.395s. The running time of proof verification is linear with the number of the challenged data blocks, ranging from 0.795s to 8.685s. As shown in Fig. 6, the running time of proof generation ranges from 0.401s to 3.793s on the cloud side. From the above experiments, we can infer that the auditing computation overhead of the TPA and the cloud both linearly increases with the number of the challenged blocks. The trade-off here is that, with more challenged blocks, the result of integrity auditing is more accurate, meanwhile, the auditing work gets more cumbersome.

2) **Communication overhead**

We evaluate the communication overhead of the auditing phase in our scheme. As discussed previously, the communication overhead is mainly from the challenge overhead and proof overhead. The challenge $chal = \{i, \beta_i\}_{i \in I}$ costs $c \cdot (|s| + |p|)$, which has a linear relationship with the number c of the challenged blocks. The proof $P = \{\mu, \sigma, vk', \mathbf{c}'\}$ costs $|p| + 2|q| + |W|$, which is independent from the number c of the challenged blocks. From Fig. 7, we can see that the communication overhead of challenge message linearly increases with the number of the challenged blocks, while the communication overhead of proof message is constant.

VIII. Conclusion

In this paper, we explore how to employ fuzzy private key to realize data integrity auditing without storing private key. We propose the first practical data integrity auditing scheme without private key storage for secure cloud storage. In the proposed scheme, we utilize biometric data (e.g. fingerprint, iris scan) as user's fuzzy private key to achieve data integrity auditing without private key storage. In addition, we design a signature scheme supporting blockless verifiability and the compatibility with the linear sketch. The formal security proof and the performance analysis show that our proposed scheme is provably secure and efficient.

References

[1] H. Dewan and R. C. Hansdah, "A survey of cloud storage facilities," in *2011 IEEE World Congress on Services*, July 2011, pp. 224–231.

[2] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, Jan 2012.

[3] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 485–497, March 2015.

[4] N. Garg and S. Bawa, "Rits-mht: Relative indexed and time stamped merkle hash tree based data auditing protocol for cloud computing," *Journal of Network & Computer Applications*, vol. 84, pp. 1–13, 2017.

[5] H. Jin, H. Jiang, and K. Zhou, "Dynamic and public auditing with fair arbitration for cloud data," *IEEE Transactions on Cloud Computing*, vol. 13, no. 9, pp. 1–14, 2014.

[6] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Comput. Electr. Eng.*, vol. 40, no. 5, pp. 1703–1713, Jul. 2014.

[7] B. Wang, B. Li, and H. Li, "Knox: privacy-preserving auditing for shared data with large groups in the cloud," in *International Conference on Applied Cryptography and Network Security*, 2012, pp. 507–525.

[8] B. Wang, H. Li, and M. Li, "Privacy-preserving public auditing for shared cloud data supporting group dynamics," in *2013 IEEE International Conference on Communications (ICC)*, June 2013, pp. 1946–1950.

[9] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1167–1179, 2015.

[10] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1362–1375, June 2016.

[11] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1931–1940, Aug 2017.

[12] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol. 8, no. 2, pp. 114–121, March 2014.

[13] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165–1176, June 2016.

[14] W. Shen, G. Yang, J. Yu, H. Zhang, F. Kong, and R. Hao, "Remote data possession checking with privacy-preserving authenticators for cloud storage," *Future Generation Computer Systems*, vol. 76, no. Supplement C, pp. 136 – 145, 2017.

[15] C. Ellison and B. Schneier, "Ten risks of pki: What you're not being told about public key infrastructure," vol. 16, no. 1, 12 2000.

[16] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, Jan 2004.

[17] S. Prabhakar, S. Pankanti, and A. K. Jain, "Biometric recognition: security and privacy concerns," *IEEE Security Privacy*, vol. 1, no. 2, pp. 33–42, Mar 2003.

[18] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in cryptology—EUROCRYPT 2005*, ser. Lecture Notes in Comput. Sci. Springer, Berlin, 2005, vol. 3494, pp. 457–473.

[19] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07, 2007, pp. 598–609.

[20] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07, 2007, pp. 584–597.

[21] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptology*, vol. 26, no. 3, pp. 442–483, Jul. 2013.

[22] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *ACM Symposium on Applied Computing*, 2011, pp. 1550–1557.

[23] M. Sookhak, A. Gani, M. K. Khan, and R. Buyya, "Dynamic remote data auditing for securing big data storage in cloud computing," *Information Sciences*, vol. 380, pp. 101–116, 2017.
[24] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.
[25] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end devices in cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2572–2583, Nov 2016.
[26] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, April 2017.
[27] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric-key based proofs of retrievability supporting public verification," in *Computer Security – ESORICS 2015*. Cham: Springer International Publishing, 2015, pp. 203–223.
[28] J. Li, X. Tan, X. Chen, D. S. Wong, and F. Xhafa, "Opor: Enabling proof of retrievability in cloud computing with resource-constrained devices," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 195–205, April 2015.
[29] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *Journal of Network and Computer Applications*, vol. 82, pp. 56–64, 2017.
[30] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *2012 IEEE Fifth International Conference on Cloud Computing*, June 2012, pp. 295–302.
[31] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, no. C, pp. 130–139, Mar. 2016.
[32] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "Npp: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Transactions on Big Data*, pp. 1–1, 2017.
[33] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 92–106, Jan.-Feb. 2015.
[34] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, 2018. [Online]. Available: doi:10.1109/TDSC.2018.2829880
[35] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics & Security*, vol. 14, no. 2, pp. 331–346.
[36] Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng, and J. Hu, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 940–952, 2017.
[37] K. Takahashi, T. Matsuda, T. Murakami, G. Hanaoka, and M. Nishigaki, "A signature scheme with a fuzzy private key," pp. 105–126, 2015.
[38] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *Siam Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.
[39] P. Yang, Z. Cao, and X. Dong, "Fuzzy identity based signature with applications to biometric authentication," *Comput. Electr. Eng.*, vol. 37, no. 4, pp. 532–540, Jul. 2011.
[40] B. Dan, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.
[41] "The gnu multiple precision arithmetic library (gmp)," http://gmplib.org/.
[42] B. Lynn, "The pairing-based cryptographic library," https://crypto.stanford.edu/pbc/, 2015.

Wenting Shen received the M.S. and B.S. degrees in college of Computer Science and Technology from Qingdao University, China, in 2017 and 2014, respectively. She is currently a Ph.D. candidate in School of Mathematics in Shandong University, China. Her research interests include cloud security and big data security.

Jing Qin is a professor in School of Mathematics, Shandong University P. R. China. She received her B.S. from Information Engineering University, Zhengzhou, P. R. China in 1982 and Ph.D. degree from School of Mathematics in Shandong University, P. R. China in 2004. Her research interests include computational number theory, information security, design and analysis of security about cryptologic protocols. She has co-authored 2 books and has published about 30 professional research papers. Prof. Qin is a senior member of Chinese Association for Cryptologic Research (CACR) as well as China Computer Federation (CCF).

Jia Yu is a professor of the College of Computer Science and Technology at Qingdao University. He received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, in 2006. He was a visiting professor with the Department of Computer Science and Engineering, the State University of New York at Buffalo, from Nov. 2013 to Nov. 2014. His research interests include cloud computing security, key evolving cryptography, digital signature, and network security.

Rong Hao works in the College of Computer Science and Technology, Qingdao University. Her research interest is cloud computing security and cryptography.

Jiankun Hu a full professor of cyber security, School of Engineering and IT, University of New South Wales, Canberra, Australia. His main research interest is in the field of cyber security, including biometrics security, where he has publications at top venues including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He has served on the editorial boards of up to seven international journals including the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.

Jixin Ma obtained his BSc and MSc of Mathematics in 1982 and 1988, respectively, and PhD of Computer Sciences in 1994. He is a Reader of Computer Science in the Department of Computing and Information Systems, and the Director of the Centre for Computer and Computational Science, as well as the Leader of Artificial Intelligence Research Group, at University of Greenwich, United Kingdom. His main research areas include Artificial Intelligence, Software Engineering and Information Systems, with special interest in Information Security..