Fine-Grained and Controllably Editable Data Sharing With Accountability in Cloud Storage

Huiying Hou, Jianting Ning, Yunlei Zhao, and Robert H. Deng, *Fellow, IEEE*

Abstract—With the increasing cloud storage service, users can enjoy non-interactive data sharing. Nonetheless, the data owner cannot timely update the shared data all the while. To ensure the timeliness and the authoritative source of the data, some users should be allowed to update the data on behalf of an authoritative data owner without changing data source. However, this allows harmful information to be injected into the data unnoticeably. How to efficiently realize editable cloud-based data sharing supporting malicious user tracing has not been fully explored. To address the problem, we propose a fine-grained and controllably editable cloud-based data sharing scheme with malicious user accountability. The data owner only needs to sign the shared data before uploading it and can specify a fine-grained access control policy about who can update the data and which portions of the data can be updated. The authorized users non-interactively convert signatures of original data into new ones for the updated data, which are indistinguishable from the original signatures. The proposed scheme also supports malicious user accountability in the sense that malicious users who post harmful information can be traced. We demonstrate the security and practicality of our scheme via formal security analysis and extensive experiments.

Index Terms—Cloud storage, data sharing, accountability, attribute-based cryptography, sanitizable signature

1 Introduction

Cloud storage is an important industry trend whereby a cloud service provider offers adequate storage resources to host its users' data. As data explodes, users are generating more data than they can store locally. Therefore, a growing number of people prefer to store their data in the external cloud [1]. Data sharing, which is one of the most basic services of cloud storage, allows users to share data with others. When a user stores data in the remote cloud, such as Google Drive, Dropbox and iCloud, this data is usually shared among multiple users (unless in a private cloud) [2]. By using or modifying the data shared by others, users may gain some profit. Moreover, cloud storage allows users to obtain the desired data anytime and anywhere, which may be owned by themselves or shared by others, bringing enormous convenience to people's life.

Despite the advantages aforementioned, cloud-based data sharing poses numerous security challenges. In most of existing cloud-based data sharing schemes, the shared data can only be updated by the data owner. Unfortunately, the data owner cannot timely update the shared data all the while. Thus, to ensure the timeliness of the data, cloud users other than the data owner should be allowed to update it on behalf of the data owner. However, this allows the incorrect or even harmful information can be injected into the shared data by malicious users. For example, an authoritative research institution sends a report on the current economic problem to an external cloud for public access. The data in the report may change over time. The research institution cannot timely update the data all the while. To ensure the timeliness and the authoritative source of the shared data, the research institution hopes that other researchers in the same research filed will be able to update the data without changing the source of the report. In this case, however, the incorrect or even harmful information may be injected into the shared data by malicious cloud users, which can seriously mislead subsequent research on the economic problem. With the rapid development of cloud-based data sharing, this problem is becoming increasingly prevalent. Therefore, how to realize editable cloud-based data sharing supporting the malicious user tracing is an extremely important and urgent problem.

One potential solution to this problem is to sign the shared data by utilizing traditional digital signature

- *Huiying Hou is with the College of Computer Science and Technology, Fudan University, Shanghai 200433, China. E-mail: 18110240036@fudan.edu.cn.*
- *Jianting Ning is with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, Fujian 350117, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. E-mail: jtning88@gmail.com.*
- *Yunlei Zhao is with the College of Computer Science and Technology, Fudan University, Shanghai 200433, China, and also with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi 710071, China. E-mail: ylzhao@fudan.edu.cn.*
- *Robert H. Deng is with the Secure Mobile Centre, School of Information Systems, Singapore Management University, Singapore 178902, Singapore. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 12 Oct. 2020; revised 6 May 2021; accepted 21 July 2021. Date of publication 27 July 2021; date of current version 2 Sept. 2022.
This work was supported by National Natural Science Foundation of China under Grants U1536205, 61472084, 61972094, and 62032005, National Key Research and Development Program of China under Grant 2017YFB0802000, Shanghai Innovation Action Project under Grant 16DZ1100200, Shanghai Science and Technology Development Funds under Grant 16JC1400801, Shandong Provincial Key Research and Development Program of China under Grants 2017CXGC0701 and 2018CXGC0701, the young talent promotion project of Fujian Science and Technology Association and Science Foundation of Fujian Provincial Science and Technology Agency under Grant 2020J02016.
(Corresponding author: Yunlei Zhao.)
Digital Object Identifier no. 10.1109/TDSC.2021.3100401

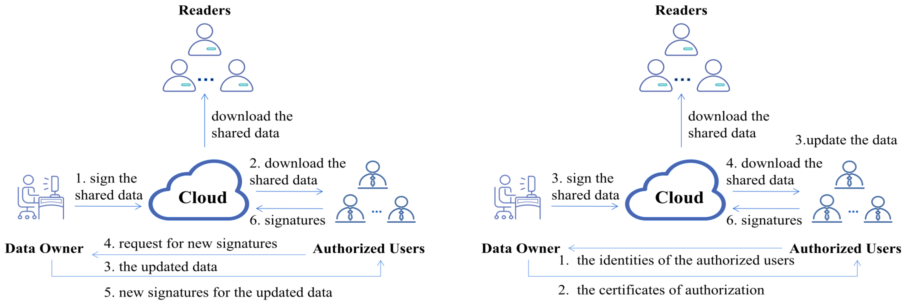

Fig. 1. The flow chart of potential solutions.

algorithms before uploading it to the cloud server. When some users want to update the shared data, as shown in Fig. 1a, they need to first interact with the original data owner, who will regenerate the new corresponding signatures for the updated data if it is valid. As mentioned above, cloud-based data sharing allows users to access the shared data as needed anytime and anywhere. Therefore, in order to generate the new corresponding signatures for the updated data, the data owner of the shared data must be always online. Clearly, such an approach is impractical. This due to several ineluctable reasons: 1) the data owner may be temporarily disconnected due to inevitable hardware faults or software bugs; 2) the network servers may temporarily goes down due to some mechanical faults; 3) the data owner may be subject to internal or external attacks.

Another potential approach is to sign the shared data by utilizing proxy digital signature algorithms. In this way, as shown in Fig. 1b, the data owner needs to know the identities of the candidate authorized users in advance to send the certificate of authorization to them. However, the number and exact identities of the candidate authorized users is uncertain and cannot be known in advance generally. For example, the number of researchers in the same research field is constantly changing around the world. Besides, not all parts of the shared data are allowed to be updated by others. For instance, the results of simulation experiments in the shared report cannot be updated because it directly determines whether the research findings are correct. Therefore, how to efficiently realize fine-grained and controllably editable data sharing with accountability in cloud storage is very important and valuable. Unfortunately, this problem has not been fully explored.

1.1 Contribution

In a nutshell, this paper mainly has the following contributions:

1) We investigate the above interesting problems and propose a fine-grained and controllably editable data sharing scheme with accountability in cloud storage. In this scheme, when data owners upload data to the cloud, they can design a fine-grained access control policy that specifies who can update the data and which portions of the data can be updated. Only authorized users can update the portions of the data that are allowed to be updated. The scheme supports the malicious user accountability, which can distinguish between the responsibility of the data owner and that of the authorized users. In this case, the data owner cannot accuse the authorized users (vice versa) of signing. In addition, the authorized users can non-interactively convert signatures of original data into new ones for the updated data. These new signatures are indistinguishable from the original signatures generated by the data owner.
2) We design a novel attribute-based sanitizable signature as the underlying technology to support the fine-grained and controllably editable data sharing scheme with accountability in cloud storage. In such an attribute-based sanitizable signature, the data owner can have fine-grained control over which parts of the data can be updated and who can update the signed data without knowing the number and exact identities of the authorized sanitizers. Even authorized sanitizers can only update the parts of the data that are allowed to be updated. In addition, the signature allows a trust authority to trace the exact identity of the signer (the original data owner or the sanitizer).
3) We present the formal security analysis for the proposed scheme and evaluate its performance through extensive experiments, which demonstrate that the proposed scheme is secure and efficient.

1.2 Related Work

The concept of sanitizable signature was introduced by Ateniese *et al.* [3] first. A sanitizable signature scheme allows a sanitizer to update the signed data allowed to be updated and generate the new corresponding signature for the updated data without interacting with the original signer. In order to ensure the security of the scheme, two necessary security requirements are defined in their scheme: (1) unforgeability, that is, only authorized sanitizers can generate the new valid signatures for the updated data. (2) Transparency, that is, the updated data and its signatures are indistinguishable from the original information and corresponding signatures. Unfortunately, they did not give a complete definition of the sanitizable signature, nor did they provide the formal security analysis. Brzuska *et al.* [4], [5] provided a perfect formalized definition of the sanitizable signature and gave a formalized definition of the basic security requirements. They introduced five formal security requirements: unforgeability, immutability, privacy, transparency, accountability, and analyzed the relationships between these security requirements. Canard *et al.* [6] proposed a generic construction of the of trapdoor sanitizable signature. In this scheme, the sanitizer can generate the valid signature for the updated data after receiving the trapdoor key from the original signer. Using an accountable chameleon hash, Lai *et al.* [7] proposed an accountable trapdoor sanitizable signature. However, neither of the above two schemes gives the concrete construction of the sanitizable signature. After that, many concrete sanitizable signature schemes are proposed [8], [9], [10]. Bultel *et al.* [9] proposed an invisible and unlinkable sanitizable signature, which efficiently achieves invisibility and unlinkability simultaneously. Xu *et al.* [10] presented a sanitizable signature, which is used to achieve privacy-preserving for smart mobile medical scenarios. We naively try to use one of the above sanitizabe signature schemes to realize editable cloud-based data sharing system. Since none of the aforementioned schemes simultaneously supports fine-grained control over candidate sanitizer and the malicious user accountability, all of them are not suitable for cloud-based data sharing environments.

Attribute-based cryptography schemes can provide fine-grained access control. Generally, attribute-based cryptography is divided into three types: attribute-based encryption (ABE) [11], [12], [13], attribute-based signature (ABS) [14], and attribute-based signcryption (ABSC) [15]. Attribute-based signature is an extension of the attribute-based encryption. Maji *et al.* [14] proposed an attribute-based signature scheme with an expressive access structure, which only proved security in the general group model. Li *et al.* [16] later proposed two efficient ABS schemes in the random oracle model and the standard model respectively. However, both constructions support threshold predicate, which is not an expressive access structure. Okamoto and Takashima [17] proposed an ABS scheme which supports non-monotone access structure. They gave the formal security analysis in the standard model. To support flexible access structure, two ABS schemes in random oracle model and standard model are proposed in [18], [19] respectively. Li *et al.* [20] proposed a multi-authority ABS scheme which supports threshold gates.

The aforementioned attribute-based signature schemes do not allow a user to update the data and generate the new valid signature for it without interacting with the original signer. In order to address this problem, some attribute-based sanitizable signature schemes are proposed [21], [22], [23], [24]. The scheme in [22] did not give the specific construct of the attribute-based sanitizable signature. The scheme in [21] did not support the expressive access structure. The scheme in [23] only provided an all-or-nothing solution for data modification. The number of blocks of the signed data cannot be changed and the set of inadmissible blocks needs to be stored in [24]. In a real environment of cloud-based data sharing, the data owner should have fine-grained control over who can update the shared data without knowing the exact members and number of data consumers, and specify which portions of the shared data can be updated. However, the above schemes are not applicable to realizing editable cloud-based data sharing with the malicious user accountability. In this paper, we explore how to realize editable cloud-based data sharing with accountability, and design an attribute-based sanitizable signature which supports malicious users tracing and allows the data owner has fine-grained control over who can update the shared data and specify which portions of the shared data can be updated.

TABLE 1
Notations

Notation	Meaning
p	One large prime
G, G_1, G_T	Multiplicative cyclic groups with the prime order p
g	A generator of group G
$\hat{e}$	A bilinear map $\hat{e} : G \times G_1 \rightarrow G_T$
Z_p^*	A prime field with nonzero elements
λ	The security parameter
msk	The master private key
mpk	The master public key
S_{DW}	The set of attributes owned by the data owner
dID	The data owner's identity
pk_{DW}	The public signing key
sk_{DW}	The private signing key
$Ask_{dID,S_{DW}}$	The secret attribute key for data owner
S_{AU}	The set of attributes owned by an authorized user
aID	The authorized user's identity
$Ask_{aID,S_{AU}}$	The secret attribute key for the authorized user
m	The message
P	The access policy
am	The description of the admissible modification
σ	The signature
dm	The description of the desired modification
σ'	The new valid signature
tk	The tracing key

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we briefly review the preliminary knowledge related to this paper. We give the definition of system model and security model in Section 3. In Section 4, we describe the proposed scheme in detail. In Section 5, we introduce the formal proof of the security of the proposed scheme. We evaluate the performance of the proposed scheme in Section 6. Finally, we come to the conclusion in Section 7.

2 Preliminaries

2.1 Notions

We list the notations used in our scheme in Table 1.

2.2 Prime Order Bilinear Groups

Let $\Phi(1^\lambda)$ be an algorithm, which takes security parameter λ as input and outputs a symmetric bilinear map of the prime order p. Let $(p, G, G_1, G_{\mathrm{T}}, \hat{e})$ denote the output of the algorithm $\Phi(1^\lambda)$, where G, G_1 and G_{T} are three multiplicative cyclic groups with the prime order p. The bilinear map $\hat{e} : G \times G_1 \rightarrow G_T$ satisfies the following characteristics:

- *Bilinearity*: $\hat{e}(u^a, v^b) = \hat{e}(u^b, v^a) = \hat{e}(u, v)^{ab}$ for $\forall u \in G$, $\forall v \in G_1$ and $\forall a, b \in \mathrm{Z}_p$.
- *Non-degeneracy*: $\hat{e}(u, v) \neq 1$.

If operations in group G, G_1 and bilinear map $\hat{e}: G \times G_1 \rightarrow G_T$ can be efficiently computed, then the group G is said to be a bilinear group.

2.3 Class-Hiding Groups

Let $\hat{e}: G \times G_1 \rightarrow G_T$ a bilinear map, where G, G_1, and G_T are three multiplicative cyclic groups with the prime order p. We set $\bar{N} = (N_1, N_2, \ldots, N_l) \in G^l$ and $\rho \in Z_p$. Let $\bar{N} := (N_1, N_2, \ldots, N_l)^\rho := (N_1^\rho, N_2^\rho, \ldots, N_l^\rho)$. Then, an equivalence relation is defined as follows:

$$R := \{(\bar{X}, \bar{Y}) : \exists l > 1, \rho \in Z_p^* s.t. (\bar{X}, \bar{Y}) \in G^l \times G^l \wedge \bar{Y} = \bar{X}^\rho\}.$$

Thus, the equivalence class of $\bar{X}$ is

$$[\bar{X}]_R := \{\bar{Y} \in G^l : (\bar{X}, \bar{Y}) \in R\}.$$

Below, we give the definition of the class hiding, which means that the elements from the same equivalence class and randomly sampled group are indistinguishable.

Definition 1 (Class-Hiding). *If, for all $l > 1$, the probability of PPT adversaries $\mathcal{A}$ distinguishing the elements from same equivalence class and randomly sampled is negligible, we say that the relationship $\mathcal{R}$ is class-hiding. The formal definition is as follows:*

$$\left| \Pr\left[b' = b : \begin{array}{c} b \leftarrow \{0,1\}; (\bar{X}, \bar{X}_0) \leftarrow (G^l)^2 \\ \bar{X}_1 \leftarrow [\bar{X}]_R; b' \leftarrow \mathcal{A}\{\bar{X}, \bar{X}_b\} \end{array} \right] - \frac{1}{2} \right| \leq negl(\lambda),$$

where $negl(\lambda)$ denotes a negligible function.

Lemma 1 ([25]). *A relation $\mathcal{R}$ is said to be class-hiding if and only if the Decisional Diffie-Hellman (DDH) assumption holds in G_1.*

The above lemma has been proved in [25].

2.4 Equivalence Class Signatures

As defined in [25], the equivalence class signature algorithm allows the user to sign a element of the equivalence class defined above, which can be updated to a new signature of the random element in the same equivalence class. The formal definition of the equivalence class signature is as follows:

Definition 2 (Equivalence Class Signatures). *An equivalence class signature (EQS) contains the following five algorithms:*

- $(pk, sk) \leftarrow KGen(\hat{e}, 1^l)$: *This is the key generation algorithm. It takes the bilinear map $\hat{e}$ and the message length* l *($l > 1$) as input, and outputs the private/public key pair (pk, sk).*
- $\sigma \leftarrow Sign(sk, \bar{X})$: *This is the signing algorithm. It takes the key* sk *and the message $\bar{X} \in G^l$ that need to be signed as input, and outputs the corresponding signature σ for $[\bar{X}]_R$.*
- $\sigma' \leftarrow ChgRep(pk, \bar{X}, \sigma, \rho)$: *This is the change representation algorithm. It takes the key* pk*, the message $\bar{X} \in G^l$, the signature σ and the scalar ρ as input, and outputs the fresh signature for $[\bar{X}^\rho]_R$.*
- $b \leftarrow Vf(pk, \bar{X}, \sigma)$: *This is the signature verification algorithm. It takes the key* pk*, the message $\bar{X} \in G^l$, the signature σ as input, and outputs $b = 1$ if σ is valid. Otherwise, $b = 0$.*
- $b \leftarrow VfKey(pk, sk)$: *This is the key verification algorithm. It takes the key* pk *and* sk *as input, and outputs $b = 1$ if the keys are consistent. Otherwise, $b = 0$.*

The detailed definition of correctness and formal security proof of the equivalence class signature (EQS) are given in [25].

2.5 Monotone Span Program

The triple $\mathcal{M} = (F, M, f)$ denotes a monotone span program, where F is a field, M is an $a \times b$ matrix over the field F, and f is a map $\{1, \ldots, a\} \rightarrow \{p_1, \ldots, p_n\}$, where $p_i, i \in \{1, \ldots, n\}$ denotes a user. Let M_A denote the sub-matrix of M, which contains the rows mapped to A ($A \subseteq \{p_1, \ldots, p_n\}$). If the rows of M_B span the vector $(1, 0, \ldots, 0)$, the B is said to be accepted by the $\mathcal{M}$. If $\forall B \in$ T can be accepted by $\mathcal{M}$, the access structure T can be accepted by $\mathcal{M}$.

Example. Given the monotone span program (F_{17}, M, f) as follows:

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}$$

$f(1) = f(2) = p_2$, $f(3) = p_1$ and $f(4) = p_3$. Let $A = \{p_1, p_3\}$ and $B = \{p_1, p_2\}$. Thus

$$M_A = \begin{pmatrix} 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}, M_B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix}.$$

From the above we know that M_B has full rank, and $(3, 14, 1)M_B = (1, 0, 0)$. Therefore, $\mathcal{M}$ accepts the set $B = \{p_1, p_2\}$. However, the rows of M_A do not span the vector $(1, 0, \ldots, 0)$, the set $A = \{p_1, p_3\}$ cannot be accepted by the $\mathcal{M}$.

In addition, a monotone span program implies a linear secret-sharing scheme. For example, consider the shared secret is $k \in F$. Then, we randomly select $(r_2, \ldots, r_b) \leftarrow F$, and set $\mathbf{r} = (k, r_2, \ldots, r_b)$. Let $M\mathbf{r} = (s_1, \ldots, s_a)$, and distribute the shares to each corresponding p_i. As mentioned above, $(3, 14, 1)M_B = (1, 0, 0)$. Let $v = (3, 14, 1)$, we have $v(M_B\mathbf{r}) = (vM_B)\mathbf{r} = (1, 0, 0)\mathbf{r} = k$. The detailed proof of these is given in [26].

2.6 Traceable Attribute-Based Signatures

The traceable attribute-based signatures allows the message signed by the user whose attributes satisfying the signing policy, and allows the trust authority to recover the exact identity of the signer. The detailed definition of traceable attribute-based signatures is as follows:

Definition 3 (Traceable Attribute-Based Signatures). *A traceable attribute-based signature (TABS) contains the following five algorithms:*

- $Setup(1^{\lambda})$: *This algorithm takes the security parameter* λ *as input, and outputs the public system parameter* pp, *the master secret key* msk *and the tracing key* tk.
- $KeyGen(pp, uID, msk, S)$: *This algorithm takes the public system parameter* pp, *the identity of the user* uID, *the master secret key* msk, *and the set of attributes* S *as input, and outputs the secret key* $sk_{uID,S}$.
- $Sign(pp, sk_{uID,S}, m, P)$: *This algorithm takes the public system parameter* pp, *the secret key* $sk_{uID,S}$, *the message* m *and a signing policy* P *as input, and outputs the signature* σ.
- $Verify(pp, m, \sigma, P)$: *This algorithm takes the public system parameter* pp, *the message* m, *the signature* σ *and the signing policy* P *as input, and outputs a bit* b. *If the signature is valid, then* $b = 1$. *Otherwise,* $b = 0$.
- $Trace(tk, \sigma, pp)$: *This algorithm takes the tracing key* tk, *the signature* σ, *and the public system parameter* pp *as input, and outputs user's identity* uID.

The detailed definition of correctness and formal security proof of the traceable attribute-based signature (TABS) are given in [27].

2.7 Ciphertext-Policy Attribute-Based Encryption

In the ciphertext-policy attribute-based encryption (CP-ABE) scheme, the ciphertext is attached to a access policy, and the decryption key is associated with a set of attributes. The ciphertext can be decrypted correctly only if the decryption key owned by the user satisfies the access policy. The detailed definition of the ciphertext-policy attribute-based encryption (CP-ABE) is as follows:

Definition 4 (Ciphertext-Policy Attribute-Based Encryption). *A ciphertext-policy attribute-based encryption contains the following four algorithms:*

- $Setup(1^{\lambda})$: *This algorithm takes the security parameter* λ *as input, and outputs the public parameter* pp *and the master secret key* msk.
- $Encrypt(pp, P, m)$: *This algorithm takes the public parameter* pp, *the access policy* P *and the message* m *as input, and outputs the corresponding ciphertext* C.
- $KeyGen(msk, S)$: *This algorithm takes the master secret key* msk *and the set of attributes* S *as input, and outputs the secret key* sk *for the set of attributes* S.
- $Decrypt(pp, C, sk)$: *This algorithm takes the public parameter* pp, *the corresponding ciphertext* C *and the secret key* sk, *and outputs the message* m'.

The detailed definition of correctness and formal security proof of the ciphertext-policy attribute-based encryption (CP-ABE) are given in [28].

3 Problem Formulation

3.1 System Model

The system model of the proposed scheme consists of five kinds of different entities: the Cloud, the Trust Authority (TA), the Data Owner, the Authorized Users and the Readers, as shown in Fig. 2.

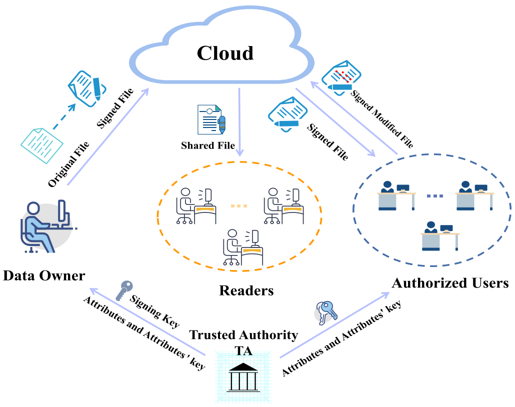

Fig. 2. The system model.

- *Cloud*. The cloud is assumed to be semi-honest. Specifically, it can only store the shared data as well as the corresponding signatures, and will not generate the new signatures for updated data as the authorized user does. In addition, the cloud has adequate storage and computing resources. With outsourcing data in a remote cloud, the users' local burden of storage and computing are remarkably reduced, and users can also share data with others non-interactively.
- *Trusted Authority*. The Trusted Authority (TA) is fully honest and responsible for generating the signing private key for the data owner, and issuing the attributes and attributes' key for the data owner and all authorized users.
- *Data Owner*. The data owner is fully honest and generates a signature for the original file before uploading the data to the cloud. The data owner has fine-grained control over which users can update the file and which portions of the file can be updated.
- *Authorized Users*. The authorized users are semi-honest in the sense that they can update the parts of the file that are allowed to be updated, and generate the new valid signatures for the updated data that are indistinguishable from the signatures that the data owner generated for the original file.
- *Readers*. The readers may act malicious behavior and can only access the shared file and cannot update it. The readers may either have access to the original file signed by the data owner or to the updated file signed by the authorized users.

First, the data owner signs the original file to generate the signed file. Meanwhile, the data owner specifies which portions of the file are allowed to be updated and an access policy that is used to fine-grained control over which users can update the shared file. Then, the data owner sends the file and the corresponding signed file to the cloud. Only the authorized users can update the parts of the signed file that is allowed to be updated. The authorized user updates the data and converts signatures of original file into new ones for the updated file, which are indistinguishable from the original signatures. Next, the authorized user uploads the updated file and the corresponding signed file to the remote cloud. Readers can access the shared file stored in the cloud anytime and anywhere. This file may be the original file signed by the data owner, or an updated file. But readers are not allowed to update the shared file. When readers find that the shared file or its signature is invalid, they can capture the exact identity of the wrong side with the help of TA.

3.2 Design Goals

To realize controllably editable cloud-based data sharing with accountability, our scheme is designed to achieve the following goals:

- *Fine-grained Access Control*. Without knowing the number and exact identities of the potential authorized users, the data owner gets fine-grained control over who can update the shared file that are uploaded to the external cloud.
- *Controllable Edit*. The authorized users cannot update the portions of the shared file that are not allowed to be updated.
- *Transparency*. The authorized users' signatures of the updated file is indistinguishable from the data owner's signatures for the original file.
- *Accountability*. The data owner cannot accuse the authorized users (vice versa) of signing.

3.3 Definitions

3.3.1 Scheme Definitions

Definition 5 (Fine-Grained and Controllably Editable Data Sharing Scheme With Accountability in Cloud Storage). *A fine-grained and controllably editable data sharing scheme with accountability in cloud storage consists of the following seven algorithms:* Setup, KGenDW, KGenAU, Sign, SignChg, Verify, Trace. *The above algorithms are described as follows in detail:*

- $(mpk, msk, tk) \leftarrow Setup(1^\lambda, 1^l)$: *The set up algorithm is run by TA and takes the security parameter 1^λ and the maximum length 1^l of the messages as input. It outputs the master private/public key pair (mpk, msk) and the tracing key* tk.
- $(pk_{DW}, sk_{DW}, Ask_{dID,S_{DW}}) \leftarrow KGenDW(mpk, msk, S_{DW}, dID)$: *The data owner's key generation algorithm is run by TA and takes the master private/public key pair (mpk, msk), the set of attributes S_{DW} owned by the data owner and the data owner's identity dID as input. It outputs the private/public signing key pair (pk_{DW}, sk_{DW}) and the secret attribute key $Ask_{dID,S_{DW}}$ for the data owner.*
- $(Ask_{aID,S_{AU}}) \leftarrow KGenAU(mpk, msk, S_{AU}, aID)$: *The authorized user's key generation algorithm is run by TA and takes the master private/public key pair (mpk, msk), the set of attributes S_{AU} owned by an authorized user and the authorized user's identity aID as input. It outputs the secret attribute key $Ask_{aID,S_{AU}}$ for the authorized user.*
- $\sigma \leftarrow Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am)$: *The signature generation algorithm is run by the data owner and takes the master public key mpk, the message* m*, the access policy P, the data owner's signing key sk_{DW}, the data owner's attribute key $Ask_{dID,S_{DW}}$ and the description* am *of the admissible modification as input. It outputs the signature σ.*
- $\sigma' \leftarrow SignChg(mpk, pk_{DW}, m, P, \sigma, dm, Ask_{aID,S_{AU}})$: *The signature change algorithm is run by the authorized users and takes the master public key mpk, the data owner's public key pk_{DW}, the message* m*, the access policy P, the original signature σ, the description* dm *of the desired modification, the authorized user's attribute key $Ask_{aID,S_{AU}}$ as input. It outputs the new valid signature σ'.*
- $b \leftarrow Verify(mpk, pk_{DW}, P, m, \sigma)$: *The verification algorithm can run by anyone and takes the master public key mpk, the data owner's public key pk_{DW}, the access policy P, the message* m *and the signature σ as input. It outputs a bit* b*. If the signature is valid, $b = 1$. Otherwise, $b = 0$.*
- $dID/aID \leftarrow Trace(mpk, \sigma, tk, st)$: *The trace algorithm is run by TA and takes the master public key mpk, the signature σ, the tracing key* tk *and the list* st *stored in TA as input. It outputs the data owner's identity* dID*, or an authorized user's identity* aID.

3.3.2 Security Definitions

Definition 6 (Controllable Edit). *In order to formally describe the controllable edit of the shared data, we introduce a game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ to show how the adversary $\mathcal{A}$ is against the controllable edit of the shared data. Trusted authority is viewed as a challenger $\mathcal{C}$ and the authorized user is viewed as an adversary $\mathcal{A}$ in our security definition. This game includes the following phases:*

- Setup Phase: *First, the challenger $\mathcal{C}$ runs the* Setup *algorithm to generate the master private/public key pair (mpk, msk) and the tracing key* tk*. Then, $\mathcal{C}$ holds the master private key msk and the tracing key* tk *locally. Finally, $\mathcal{C}$ sends the master public key mpk to the adversary $\mathcal{A}$.*
- Query Phase:
  - KGenDW Queries*: The adversary $\mathcal{A}$ makes queries the data owner's private/public key pair and the attribute key for the set of attributes S'_{DW} and the identity dID'. $\mathcal{C}$ runs* KGenDW *algorithm and returns the private/public key pair (pk'_{DW}, sk'_{DW}) and the attribute key $Ask_{dID',S'_{DW}}$ to $\mathcal{A}$.*
  - KGenAU Queries*: The adversary $\mathcal{A}$ makes queries the attribute key of the potential authorized user for the identity aID' of the potential authorized user and the attributes' set S'_{AU}. $\mathcal{C}$ runs* KGenAU *algorithm and returns the attribute key $Ask_{aID',S'_{AU}}$ to $\mathcal{A}$.*
  - Sign Queries*: The adversary $\mathcal{A}$ makes queries the signature for the message* m*, the data owner's signing key sk'_{DW}, the attribute key $Ask_{dID',S'_{DW}}$ and the description am' of the modification. $\mathcal{C}$ runs* Sign *algorithm and returns the signature σ' to $\mathcal{A}$.*
  - SignChg Queries*: The adversary $\mathcal{A}$ makes queries the new signature for the message* m*, the access policy P, the signature σ', the description dm' of desired modification and the potential authorized user's attribute key $Ask_{aID',S'_{AU}}$. $\mathcal{C}$ runs* SignChg *algorithm and returns the signature σ'' to $\mathcal{A}$.*
  - Verify Queries*: The adversary $\mathcal{A}$ makes queries the verification result for data owner's public key pk'_{DW}, the message* m *and the signature σ'. $\mathcal{C}$ runs* Verify *algorithm and returns the result to $\mathcal{A}$.*
  - Trace Queries*: The adversary $\mathcal{A}$ makes queries the signer's identity for the signature σ', the*

tracing key tk *and the information* st *stored in TA.* $\mathcal{C}$ *runs* Trace *algorithm and returns the signer's identity to* $\mathcal{A}$.

- Challenge Phase: *The adversary* $\mathcal{A}$ *adaptively chooses the authorized user's attributes set* S^*_{AU} $(P(S^*_{AU}) = 1)$ *and the identity* aID^*. *Then,* $\mathcal{A}$ *runs* SignChg *algorithm to generate the challenged signature* σ^* *with the updated data* $dm^*(m) = m^* \not\subset am'(m)$. *Finally, the adversary* $\mathcal{A}$ *sends* $(S^*_{AU}, m^*, \sigma^*)$ *to* $\mathcal{C}$.
- Verify Phase: *The adversary* $\mathcal{A}$ *performs polynomial queries as in* Query Phase. *Consider the adversary* $\mathcal{A}$ *has made L queries, and let* $Q = \{pk_{DW,i}, S_{AU,i}, m_i, am_i, \sigma_i\}_{i=1}^{[|Q|]}$ *denote the set of information obtained through these queries.* $\mathcal{C}$ *runs* $Verify(mpk, pk'_{DW}, P, m^*, \sigma^*)$ *algorithm, and outputs a bit* b_0. *Then,* $\mathcal{C}$ *checks whether there exists a* $i \in [|Q|], dm^*(m) \subseteq am'(m)$ *such that* $S^*_{AU} = S_{AU,i}$ *and* $m^* = dm^*(m) \not\subset am'(m)$. *If there is such an* i, *the challenger* $\mathcal{C}$ *outputs* $b_1 = 1$. *Otherwise,* $\mathcal{C}$ *outputs* $b_1 = 0$.

We say that the adversary $\mathcal{A}$ *wins if* $b_0 \wedge \neg b_1 = 1$. *In the above game, we want to show that the adversary* $\mathcal{A}$*, who update the inadmissible parts of the shared data, should not generate the new valid signature. The adversary's goal is to correctly generate the valid signature* σ'' *for the inadmissible modification* $m^* = dm^*(m) \not\subset am'(m)$. *We set the advantage of a polynomial time adversary* $\mathcal{A}$ *in this game to be* $\Pr[b_0 \wedge \neg b_1 = 1]$. *We say the proposed scheme satisfies the controllable edit of the shared data if for any polynomial time adversary* $\mathcal{A}$, $\Pr[b_0 \wedge \neg b_1 = 1] < 1/poly(n)$ *for a sufficiently large* n, *where* poly *stands for a polynomial function.*

Definition 7 (Transparency). *In order to formally describe the transparency of the signature for the updated data, we introduce a game between the challenger* $\mathcal{C}$ *and the adversary* $\mathcal{F}$ *to show how the adversary* $\mathcal{F}$ *is against the transparency of the signature for the updated data. Trusted authority is viewed as a challenger* $\mathcal{C}$ *and the unauthorized user and reader are viewed as an adversary* $\mathcal{F}$ *in our security definition. This game includes the following phases:*

- Setup Phase: *First,* $\mathcal{C}$ *runs the setup algorithm to generate the master private/public key pair* (mpk, msk) *and the tracing key* tk. *Then,* $\mathcal{C}$ *holds the master private key* msk *and the tracing key* tk *locally. Finally,* $\mathcal{C}$ *sends the master public key* mpk *to the adversary* $\mathcal{F}$.
- Query Phase:
  - KGenDW Queries*: The adversary* $\mathcal{F}$ *makes queries the data owner's private/public key pair and the attribute key for the set of attributes* S'_{DW} *and the identity* dID'. $\mathcal{C}$ *runs* KGenDW *algorithm and returns the private/public key pair* (pk'_{DW}, sk'_{DW}) *and the attribute key* $Ask_{dID',S'_{DW}}$ *to* $\mathcal{F}$.
  - KGenAU Queries*: The adversary* $\mathcal{F}$ *makes queries the attribute key of the potential authorized user for the identity* aID' *of the potential authorized user and the attributes' set* S'_{AU}. $\mathcal{C}$ *runs* KGenAU *algorithm and returns the attribute key* $Ask_{aID',S'_{AU}}$ *to* $\mathcal{F}$.
  - Sign Queries*: The adversary* $\mathcal{F}$ *makes queries the signature for the message* m, *the data owner's signing key* sk'_{DW} *and the attribute key* $Ask_{dID',S'_{DW}}$, *the description* am' *of the modification.* $\mathcal{C}$ *runs* Sign *algorithm and returns the signature* σ' *to* $\mathcal{F}$.
  - SignChg Queries*: The adversary* $\mathcal{F}$ *makes queries the new signature for the message* m, *the access policy P, the signature* σ', *the description* dm' *of desired modification and the potential authorized user's attribute key* $Ask_{aID',S'_{AU}}$. $\mathcal{C}$ *runs* SignChg *algorithm and returns the signature* σ'' *to* $\mathcal{F}$.
  - Verify Queries*: The adversary* $\mathcal{F}$ *makes queries the verification result for data owner's public key* pk'_{DW}, *the message* m *and the signature* σ'. $\mathcal{C}$ *runs* Verify *algorithm and returns the result to* $\mathcal{F}$.
  - Trace Queries*: The adversary* $\mathcal{F}$ *makes queries the signer's identity for the signature* σ', *the tracing key* tk *and the information* st *stored in TA.* $\mathcal{C}$ *runs* Trace *algorithm and returns the signer's identity to* $\mathcal{F}$.
- Challenge Phase: *The adversary* $\mathcal{F}$ *adaptively chooses the authorized user's attributes set* S^*_{AU} $(P(S^*_{AU}) = 1)$, *the identity* aID^* *and the modification* $dm^*(m) = m^* \subset am'(m)$, *where the message* m *is not signed by the data owner or the authorized user. Then,* $\mathcal{C}$ *randomly selects* $b \leftarrow \{0, 1\}$, *and sets* $m_0 = m$, $m_1 = m^*$. *Finally, if* $b = 1$, *the challenger* $\mathcal{C}$ *runs* $SignChg\ (mpk, pk_{DW}, m, P, \sigma, dm^*(m) = m^* \subseteq am'(m), Ask_{aID,S^*_{AU}})$, *and runs* $Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am')$ *for* $b = 0$.
- Guess Phase: *The adversary* $\mathcal{F}$ *performs polynomial queries as in* Query Phase. *Then,* $\mathcal{F}$ *returns a bit* b'.

In the above game, we want to show that the adversary $\mathcal{F}$ cannot tell the difference between the signatures produced by the data owner and the authorized users. The adversary's goal is to correctly guess the algorithm performed by $\mathcal{C}$. We set the advantage of a polynomial time adversary $\mathcal{F}$ in this game to be $\Pr[b' = b] - \frac{1}{2}$. We say the proposed scheme satisfies the transparency of the signature for the updated data if for any polynomial time adversary $\mathcal{F}$, $|\Pr[b' = b] - \frac{1}{2}| < 1/poly(n)$ for a sufficiently large n, where *poly* stands for a polynomial function.

Definition 8 (Fine-grained Access Control). *We say a controllably editable data sharing scheme with accountability achieves fine-grained access control if the data owner can get fine-grained control over who can update the shared data without knowing the exact members and number of data consumers, and specify which portions of the shared data can be updated.*

Definition 9 (Accountability). *We say a fine-grained and controllably editable data sharing scheme supports accountability if TA can extract signer's identity from any valid signature with non-negligible probability.*

4 The Proposed Scheme

4.1 An Overview

In order to efficiently achieve fine-grained and controllably editable data sharing with the malicious user accountability in the cloud storage, we first try to adopt policy-based sanitizable signature [24]. However, the number of blocks of the

$\underline{(mpk, msk, tk) \leftarrow Setup(1^{\lambda}, 1^{l})}$
$(pp_{ABE}, msk_{ABE}) \leftarrow CP-ABE.Setup(1^{\lambda})$
$(pp_{ABS}, msk_{ABS}, tk_{ABS}) \leftarrow TABS.Setup(1^{\lambda})$
$mpk = \{pp_{ABE}, pp_{ABS}\}$
$msk = \{msk_{ABE}, msk_{ABS}\}$
return $mpk, msk, tk = tk_{ABS}$

$\underline{(pk_{DW}, sk_{DW}, Ask_{dID,S_{DW}}) \leftarrow KGenDW(mpk, msk, S_{DW}, dID)}$
$(pk_{EQS}, sk_{EQS}) \leftarrow EQS.KGen(mpk, 1^{l})$
$Ask_{dID,S_{DW}} \leftarrow TABS.KeyGen(mpk, dID, msk, S_{DW})$
return $pk_{DW} = pk_{EQS}, sk_{DW} = sk_{EQS}, Ask_{dID,S_{DW}}$

$\underline{Ask_{aID,S_{AU}} \leftarrow KGenAU(mpk, msk, S_{AU}, aID)}$
$ABS.Ask_{aID,S_{AU}} \leftarrow TABS.KeyGen(mpk, msk, S_{AU}, aID)$
$ABE.Ask_{aID,S_{AU}} \leftarrow ABE.KeyGen(mpk, msk, S_{AU})$
return $Ask_{aID,S_{AU}} = \{ABS.Ask_{aID,S_{AU}}, ABE.Ask_{aID,S_{AU}}\}$

$\underline{b \leftarrow Verify(mpk, pk_{DW}, P, m, \sigma)}$
$b_{-3} = TABS.Verify(mpk, m, \sigma, P), b_{-2} = (\forall i \in [l], Y_i \neq g)$
$b_{-1} = EQS.Vf(pk_{EQS}, (X_1, ..., X_l), \mu), b_0 = EQS.Vf(pk_{EQS}, (Y_1, ..., Y_l), \eta)$
$b_i = (\hat{e}(X_i, \sigma_i) = \hat{e}(Y_i, H(i||m_i))), \forall i \in [l]$
return $\bigcap_{b=-3}^{l} b_i$

$\underline{dID/aID \leftarrow Trace(mpk, \sigma, tk, st)}$
$dID/aID \leftarrow TABS.Trace(mpk, \sigma, tk)$

$\underline{\sigma \leftarrow Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am)}$
if $am \neq 1$ then return $\perp$
$x_i, y_i \leftarrow Z_p^*, X_i = g^{x_i}, Y_i = X_i^{y_i}, \forall i \in [l]$
$\mu \leftarrow EQS.Sign(sk_{EQS}, (X_1, ..., X_l))$
$\eta \leftarrow EQS.Sign(sk_{EQS}, (Y_1, ..., Y_l))$
$\sigma_i = H(i||dm(m_i))^{y_i}, \forall i \in [l]$
$\xi_i = \begin{cases} y_i, i \in am \\ 0, \text{otherwise} \end{cases}$
$C = CP-ABE.Enc(mpk, P, am, \{\xi_i\}_{i\in[l]})$
$\sigma_{ABS} \leftarrow TABS.Sign(mpk, sk_{dID,S_{DW}}, m, P)$
$\sigma = \{\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^{l}, C, \sigma_{ABS}\}$

$\underline{\sigma' \leftarrow SignChg(mpk, pk_{DW}, m, P, \sigma, dm, Ask_{aID,S_{AU}})}$
if $dm \notin am$ then return $\perp$
$m' = dm(m), \quad r, s \leftarrow Z_p^*$
$(X'_1, ..., X'_l) = (X_1, ..., X_l)^r, (Y'_1, ..., Y'_l) = (Y_1, ..., Y_l)^{rs}$
$\bar{X} = (X'_1, ..., X'_l), \bar{Y} = (Y'_1, ..., Y'_l)$
$\mu' = EQS.Chg\text{Rep}(pk_{EQS}, \bar{X}, \mu, r)$
$\eta' = EQS.Chg\text{Rep}(pk_{EQS}, \bar{Y}, \eta, s)$
for each $i \in [l]$
$\xi'_i = s \cdot \xi_i$
$\sigma'_i = \begin{cases} H(i||m'_i)^{\xi'_i}, i \in am \\ \sigma_i^s, \quad \text{otherwise} \end{cases}$
endfor
$C' = CP-ABE.Enc(mpk, P, am, \{\xi'_i\}_{i\in[l]})$
$\sigma'_{ABS} \leftarrow TABS.Sign(mpk, sk_{aID,S_{AU}}, m', P)$
$\sigma' = \{\mu', \eta', \{\sigma'_i, X'_i, Y'_i\}_{i=1}^{l}, C', \sigma'_{ABS}\}$

Fig. 3. The generic construction of the proposed scheme.

signed data cannot be changed and the set of inadmissible blocks needs to be stored in [24].

Finally, we consider adopting the idea of the recent work [9], which allows the authorized users to directly generate new valid signatures on the updated data without interacting with the data owner. However, the following problems will arise if one adopts the idea of [9] directly. First, the scheme in [9] does not allow the data owner to have fine-grained control over the potential authorized users. In order to authorize the user, the description of admissible modification is required to be encrypted using the public key encryption algorithm under the public key of the desired authorized user. In this way, the data owner needs to collect the desired authorized users' public keys in advance. In most editable cloud-based data sharing environments, the data owner cannot know the number and exact identities of the potential authorized users. For example, to ensure the timeliness and the authoritative source of the shared data, an authoritative research institution hopes that other researchers in the same research filed will be able to update the data without changing the source of the report. However, the number of researchers in the same research field is constantly changing. It is difficult to count the number and identities of these researchers worldwide. Second, to achieve the malicious user accountability, verifiable ring signature [29] is adopted in the scheme [9]. The ring signature scheme requires the public keys of all potential authorized users to be known in advance, which cannot be satisfied in the cloud-based data sharing environment.

In order to solve the above problems, we improve the scheme in [9] and design a new attribute-based sanitizable signature scheme. First, the public key encryption algorithm used in the scheme [9] should be substituted with the ciphertext-policy attribute-based encryption (CP-ABE). This allows the data owner to get fine-grained control over who can update the shared data without knowing the number and exact identities of the potential authorized users. In terms of practical CP-ABE instantiations, we considered the efficient CP-ABE scheme called FAME [28]. FAME only achieves IND-CPA security. To achieve IND-CCA2 security, we convert FAME by using a variant of Fujisaki-Okamoto transform [30]. Basically, the encryption algorithm will encrypt (m, r), where *m* is the original message that needs to be encrypted, and *r* is a random string. Then, the hash value $H(r, P)$ is calculated, where H is a collision resistant hash function and P is the access policy which is contained in the ciphertext. In the decryption process, the decryption algorithm is first used to get (m', r'), and then $H'(r', P)$ is calculated. If $H' = H$, the algorithm outputs m'. Otherwise, it outputs $\perp$. Moreover, to achieve accountability on the condition that the number and exact identities of the potential authorized users are unknown, we replace the ring signature used in [9] with the traceable attribute-based signature (TABS) [27]. Our generic construction as shown in Fig. 3.

4.2 Description of the Proposed Scheme

The proposed scheme consists of the following seven algorithms:

- $(mpk, msk, tk) \leftarrow Setup(1^{\lambda}, 1^{l})$: The goal of this algorithm is to generate the tracing key *tk*, the system public parameter *mpk* and the master private key *msk* which are necessary for the subsequent algorithms.
  – On input the security parameter λ and the maximum length of the messages l, TA operates as follows. Let $\hat{e} : G \times G \rightarrow G_T$ be a bilinear pairing, where G and G_T are groups of a order n ($n = p \cdot q$, where p and q are two prime numbers), and g, h are generators of group G. Then, TA randomly selects $(a_1, a_2, b_1, b_2) \leftarrow Z_p^*, (d_1, d_2, d_3, d_4) \leftarrow Z_p$, and chooses random $\omega \in G_p$ where G_p

is the subgroup of the G of order n. Let H_1 and H_2 be two cryptographic hash functions $\{0,1\}^* \rightarrow G$.

– TA selects an automorphic signature scheme, and sets the corresponding private/public key pair to (sk_{aut}, pk_{aut}). Next, TA calculates the tracing key *tk* such that $tk = 0 \bmod p$ and $tk = 1 \bmod q$. Let the universe of attributes is U.

– Finally, TA outputs the public system parameter $mpk = (n, G, G_T, \hat{e}, g, h, \omega, H_1, H_2, g^{d_4}, pk_{aut}, U, h^{a_1}, h^{a_2}, T_1, T_2)$, where $T_1 = \hat{e}(g,h)^{d_1 \cdot a_1 + d_3}$, $T_2 = \hat{e}(g,h)^{d_2 \cdot a_2 + d_3}$ and holds the master secret key $msk = (a_1, a_2, b_1, b_2, d_4, g^{d_1}, g^{d_2}, g^{d_3}, sk_{aut}, tk)$ locally.

- $(pk_{DW}, sk_{DW}, Ask_{dID,S_{DW}}) \leftarrow KGenDW(mpk, msk, S_{DW}, dID)$: The goal of this algorithm is to generate the private/public signing key pair (pk_{DW}, sk_{DW}) and the secret attribute key $Ask_{dID,S_{DW}}$ for the data owner.
  – On input the public system parameter *mpk*, TA randomly selects $t \leftarrow Z_p^*$ and $(t_i)_{i\in[1,l]} \leftarrow (Z_p^*)^l$. Then, TA sets $sk_{EQS} = (t, (t_i)_{i\in[1,l]})$ and $pk_{EQS} = (T', (T'_i)_{i\in[1,l]} = (tg, (tgt_i)_{i\in[1,l]}))$.
  – On input the master secret key *msk*, the data owner's identity dID and attributes' set S_{DW}, TA chooses a random number $K_{dID} \in G$, and signs $K_{dID} \in G$ using the automorphic signature to get $\sigma_{K_{dID}}$. For each $at_i \in S_{DW}$, TA randomly selects $r_i \in Z_p$ and computes $sk_i = (H_1(at_i)^{d_4} K_{dID}^{r_i}, g^{r_i})$. Let $Ask_{dID,S_{DW}} = (K_{dID}, \sigma_{K_{dID}}, \{sk_i\}_{at_i \in S_{DW}})$ denote the data owner's attribute key.
  – Finally, TA sets $pk_{DW}{=}pk_{EQS}, sk_{DW} = sk_{EQS}$, $Ask_{dID,S_{DW}} = Ask_{dID,S_{DW}}$.
- $(Ask_{aID,S_{AU}}) \leftarrow KGenAU(mpk, msk, S_{AU}, aID)$: The goal of this algorithm is to generate the secret attribute key $Ask_{aID,S_{AU}}$ for the authorized user.
  – On input the master secret key *msk*, the authorized user's identity aID and attributes' set S_{AU}, TA chooses a random number $K_{aID} \in G$, and signs $K_{aID} \in G$ using the automorphic signature to get $\sigma_{K_{aID}}$. For each $at_i \in S_{AU}$, TA randomly selects $u_i \in Z_p$ and computes $ssk_i = (H_1(at_i)^{d_4} K_{aID}^{u_i}, g^{u_i})$. We set $ABS.Ask_{aID,S_{AU}} = (K_{aID}, \sigma_{K_{aID}}, \{ssk_i\}_{at_i \in S_{AU}})$ as the authorized user's attribute key for signing.
  – On input the master secret key *msk*, the authorized user's attribute set S_{AU}, TA randomly chooses $k_1, k_2 \leftarrow Z_p$ and calculates $sk_{ABE,0} = (h^{b_1 k_1}, h^{b_2 k_2}, h^{k_1 + k_2})$. For each $at_i \in S_{AU}$ and $j = 1, 2$, TA computes

$$sk_{at_i,j} = H_2(at_i||1||j)^{\frac{b_1 k_1}{a_j}} \cdot H_2(at_i||2||j)^{\frac{b_2 k_2}{a_j}} \cdot H_2(at_i||3||j)^{\frac{k_1+k_2}{a_j}} \cdot g^{\frac{r_i}{a_t}}.$$

Let $sk_{at_i} = (sk_{at_i,1}, sk_{at_i,2}, g^{-r_i})$. Then, TA calculates

$$sk'_j = g^{d_j} \cdot H_2(011j)^{\frac{b_1 k_1}{a_j}} \cdot H_2(012j)^{\frac{b_2 k_2}{a_j}} \cdot H_2(013j)^{\frac{k_1+k_2}{a_j}} \cdot g^{\frac{z}{a_j}}, z \leftarrow Z_p.$$

Let $sk' = (sk'_1, sk'_2, g^{d_3} \cdot g^{-z})$. TA outputs the authorized user's attribute key $ABE.Ask_{aID,S_{AU}} = (sk_0, \{sk_{at_i}\}_{at_i \in S_{AU}}, sk')$ for decrypting.
  – Finally, TA returns $Ask_{aID,S_{AU}} = \{ABS.Ask_{aID,S_{AU}}, ABE.Ask_{aID,S_{AU}}\}$.
- $\sigma \leftarrow Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am)$: The goal of this algorithm is to generate the signature σ.
  – On input the master public key *mpk*, the message *m*, the access policy P, the data owner's signing key sk_{DW}, the data owner's attribute key $Ask_{dID,S_{DW}}$ and the description *am* of the admissible modification, the data owner checks whether $|am| = l$. If $|am| \neq l$, the data owner returns $\perp$. Then, the data owner randomly selects $x_i, y_i \leftarrow Z_p^*, \forall i \in [l]$, and computes $X_i := g^{x_i}, Y_i := X_i^{y_i}$. The data owner calculates $\mu = EQS.Sign(sk_{EQS}, (X_1, \ldots, X_l))$ and $\eta = EQS.Sign(sk_{EQS}, (Y_1, \ldots, Y_l))$.
  – For each bit of the message *m*, the data owner computes $\sigma_i := H(i||m_i)^{y_i}$, and sets $\xi_i := \begin{cases} y_i, i \in am \\ 0, \text{otherwise} \end{cases}$.
  – On input the public system parameter *mpk*, the access policy P, the description *am* of the admissible modification and $\{\xi_i\}_{i\in[l]}$, the data owner randomly selects $s_1, s_2 \leftarrow Z_p$ and computes $ct_0 = (h^{a_1 s_1}, h^{a_2 s_2}, h^{s_1+s_2})$. Consider the monotone span program M has n_1 rows and n_2 columns. For all $u = 1, 2, \ldots, n_1$, and $\ell = 1, 2, 3$, the data owner computes

$$ct_{u,\ell} = H_2(f(i)\ell 1)^{s_\ell} \cdot H_2(f(i)\ell 2)^{s_2} \cdot \prod_{j=1}^{n_2} [H_2(0v\ell 1)^{s_1} \cdot H_2(0v\ell 2)^{s_2}]^{(M)_{u,v}},$$

where $(M)_{u,v}$ denotes $(u,v)th$ element of M. The data owner also computes $encode(am, rd)$ and $ct' = T_1^{s_1} \cdot T_2^{s_2} \cdot encode(am, rd)$, where *rd* is a random string, *encode* is an encoding algorithm. Next, the data owner computes $H_1(rd, P)$. Let $C = (ct_0, ct_1, \ldots, ct_{n_1}, ct')$.
  – On input the public system parameter *mpk*, the access policy P, the data owner's attribute key $sk_{dID,S_{DW}}$ and the message *m*, the data owner computes $\sigma_{ABS} \leftarrow TABS.Sign(mpk, sk_{dID,S_{DW}}, m, P)$. Finally, the data owner returns
$$\sigma{=}\ \{\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^l, C, \sigma_{ABS}\}.$$
- $\sigma' \leftarrow SignChg(mpk, pk_{DW}, m, P, \sigma, dm, Ask_{aID,S_{AU}})$: The goal of this algorithm is to generate the new valid signature σ'.
  – First, the authorized user runs $CP{-}ABE.Decrypt(mpk, C, ABE.Ask_{aID,S_{AU}})$ to get $encode'(am, rd)$. Then, the authorized user gets (am', rd') from $decode(encode'(am, rd))$. The data owner checks whether $dm \in am$. If $dm \notin am$, the authorized user returns $\perp$. Then, the authorized user computes $m' = dm(m)$, and selects two random number $r, s \leftarrow Z_p^*$. The authorized user calculates $(X'_1, \ldots, X'_l) = (X_1, \ldots, X_l)^r$, $(Y'_1, \ldots,$

$Y'_l) = (Y_1, \ldots, Y_l)^{rs}$ and sets $\bar{X} = (X_1, \ldots, X_l)$, $\bar{Y} = (Y_1, \ldots, Y_l)$.

 - The authorized user computes $\mu' = EQS.Chg\text{Rep}(pk_{EQS}, \bar{X}, \mu, r)$ and $\eta' = EQS.Chg\text{Rep}\$ pk_{EQS}, \bar{Y}, \eta, s)$. For all $i \in [l]$, the authorized user sets $\xi'_i = s \cdot \xi_i$ and $\sigma'_i := \{ \begin{matrix} H(i||m'_i)^{\xi'_i}, i \in am \\ \sigma_i^s, \text{otherwise} \end{matrix}$.
 - The authorized user calculates $C' := CP{-}ABE.Enc(mpk, P, am, \{\xi'_i\}_{i\in[l]})$ and $\sigma'_{ABS} \leftarrow TABS.Sign(mpk, sk_{aID,S_{AU}}, m', P)$. Finally, the authorized user returns $\sigma' = \{\mu', \eta', \{\sigma'_i, X'_i, Y'_i\}_{i\in[1,l]}, C', \sigma'_{ABS}\}$.

- $b \leftarrow Verify(mpk, pk_{DW}, P, m, \sigma)$: The goal of this algorithm is to check whether the signature is valid. It outputs a bit b. If the signature is valid, $b = 1$. Otherwise, $b = 0$. This algorithm can be performed by anyone. Computes $b_{-3} = TABS.Verify(mpk, m, \sigma, P)$, $b_{-2} = (\forall i \in [l], Y_i \neq g)$, $b_{-1} = EQS.Vf(pk_{EQS}, (X_1, \ldots, X_l), \mu)$, $b_0 = EQS.Vf(pk_{EQS}, (Y_1, \ldots, Y_l), \eta)$, and $b_i = (\hat{e}(X_i, \sigma_i) = \hat{e}(Y_i, H(i||m_i))), \forall i \in [l]$, this algorithm returns $\text{b} = \bigcap_{i=-3}^{l} b_i$.
- $dID/aID \leftarrow Trace(mpk, \sigma, tk, st)$: The goal of this algorithm is to capture the identity of the signer. TA runs $TABS.Trace(tk, \sigma, mpk)$ to obtain the identity of the signer in the signature σ.

5 Security Analysis

In this section, we analyze the security of our proposed scheme in term of controllable edit, transparency, fine-grained access control and accountability. The following proof uses the generic group model abstraction of Shoup [31]. Now, we first introduce two lemmas used in the proof process.

Lemma 2 (Schwartz-Zippel [32]). *Consider the $F(X_1, \ldots, X_m)$ is a non-zero polynomial of the degree $d \geq 0$ over the field $\mathbb{F}$. Then, for each random input $(x_1, \ldots, x_m)$, the probability of $F(x_1, \ldots, x_m) = 0$ is bounded from above by $\frac{d}{\mathbb{F}}$.*

Lemma 3. *Let $(G, G_T, g, h, \hat{e}, p)$ is the output of the algorithm $Setup(1^\lambda, 1^l)$, where $p > 2^\lambda$. Given $a, b, c \leftarrow Z_p$, the probability that the generic group adversary $\mathcal{A}$ on input $(g, g^a, g^b, h, h^b, h^c)$ outputs $(g^u, g^v, g^x, g^y, h^z)$ such that*

$$\begin{cases} au - x = 0 \\ bv - y = 0 \\ cy - xz = 0 \\ v \neq 0, \end{cases}$$

is negligible.

Proof. Suppose $(g^u, g^v, g^x, g^y, h^z)$ is the output of the generic group adversary $\mathcal{A}$. Then, there are some coefficients $(u_1, u_a, u_b, v_1, v_a, v_b, x_1, x_a, x_b, y_1, y_a, y_b, z_1, z_b, z_c) \in Z_p$ such that

$$\begin{cases} u = u_1 + au_a + bu_b \\ v = v_1 + av_a + bv_b \\ x = x_1 + ax_a + bx_b \\ y = y_1 + ay_a + by_b \\ z = z_1 + bz_b + cz_c. \end{cases}$$

We get $-x_1 + (u_1 - x_a)a - bx_b + a^2u_a + abu_b = 0$ from $au - x = 0$. For the variables A and B, $f(A, B) = -x_1 + (u_1 - x_a)A - Bx_b + A^2u_a + ABu_b$ is a quadratic polynomial. Let f is a non-zero polynomial. According to the Lemma 2, for $a, b \leftarrow Z_p$, the upper bounded of the probability of $f(a, b) = 0$ is $2/p < 2^{1-\lambda}$ which is negligible. Thus, we can set $f(A, B) = 0$. We have $x_1 = x_b = 0$. □

To the same vein, we get $v_1 = y_b$ and $y_1 = y_a = 0$ from $bv - y = 0$. Therefore, we can write $x = ax_a$ and $y = by_b$. Suppose $cy - xz = 0$, we have $bcy_b - ax_az_1 - abx_az_b - acx_az_c = 0$. According to the Lemma 2, we can assume that $y_b = 0$. Then, we get $v = v_1 = y_b = 0$, which contradicts with the above relation $v \neq 0$.

To sum up, on input $(g, g^a, g^b, h, h^b, h^c)$, the probability that the generic group adversary $\mathcal{A}$ outputs $(g^u, g^v, g^x, g^y, h^z)$ such that

$$\begin{cases} au - x = 0 \\ bv - y = 0 \\ cy - xz = 0 \\ v \neq 0, \end{cases}$$

is negligible.

Theorem 1 (Controllable Edit). *Suppose the problem defined in Lemma 3 is hard for all generic group adversaries. In the proposed scheme, for a generic group adversary $\mathcal{A}$, who updates the signed message that did not fit the modification description* am, *it is computationally infeasible to generate a valid signature for the updated message.*

Proof. To prove this theorem, we define a game between a challenger $\mathcal{C}$ and a generic group adversary $\mathcal{A}$. □

Game 1. In the *Game 1*, both the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ perform as defined in the security definition. That is, the challenger $\mathcal{C}$ runs the *Setup* algorithm and sends the master public key mpk to the adversary $\mathcal{A}$. Then, the adversary $\mathcal{A}$ does as *Query Phase*. Next, the adversary $\mathcal{A}$ adaptively chooses the authorized user's attribute set S^*_{AU} $(P(S^*_{AU}) = 1)$ and the identity aID^*. $\mathcal{A}$ runs *SignChg* algorithm to generate the challenged signature σ^* with the updated data $dm^*(m) = m^* \not\subset am'(m)$. Finally, the adversary $\mathcal{A}$ sends $(S^*_{AU}, m^*, \sigma^*)$ to the challenger $\mathcal{C}$.

Analysis. Assume that the adversary $\mathcal{A}$ wins the *Game 1* with non-negligible probability. Then, we can construct a simulator $\mathfrak{T}$ to solve the problem defined in Lemma 3. Suppose the simulator's challenge is (g, g^a, g^b, h, h^c) received from its challenger. Then, to solve the problem defined in Lemma 3, the simulator $\mathfrak{T}$ acts like the challenger $\mathcal{C}$ in *Game 1*.

- *Setup Phase:* First, the simulator $\mathfrak{T}$ runs the *Setup* algorithm to generate the master private/public key pair (mpk, msk) and the tracing key *tk*. Then, $\mathcal{C}$ holds the master private key msk and the tracing key *tk* locally. Finally, the simulator $\mathfrak{T}$ sends the master public key mpk to the adversary $\mathcal{A}$.
- *Query Phase:*
 - *KGenDW Queries*: The adversary $\mathcal{A}$ makes queries the data owner's private/public key pair and the attribute key for the set of attributes S'_{DW} and the identity dID'. The simulator $\mathfrak{T}$ runs

KGenDW algorithm and returns the private/public key pair (pk'_{DW}, sk'_{DW}) and the attribute key $Ask_{dID',S'_{DW}}$ to $\mathcal{A}$.

- *KGenAU Queries*: The adversary $\mathcal{A}$ makes queries the attribute key of the potential authorized user for the identity aID' of the potential authorized user and the attributes' set S'_{AU}. The simulator $\mathfrak{T}$ runs *KGenAU* algorithm and returns the attribute key $Ask_{aID',S'_{AU}}$ to $\mathcal{A}$.
- *Sign Queries*: The adversary $\mathcal{A}$ makes queries the signature for the message m, the data owner's signing key sk'_{DW} and the attribute key $Ask_{dID',S'_{DW}}$, the description am' of the modification. Let Q_1 denote the number of signing queries. Suppose $i^*, j^* \leftarrow Q_1$ are the signing queries which are attacked by the adversary $\mathcal{A}$, and $k^* \leftarrow [\ell]$ is the index of the inadmissible block that will be updated. If $i \neq i^*$ and $i \neq j^*$, the simulator $\mathfrak{T}$ runs *Sign* algorithm and returns the signature σ' to $\mathcal{A}$ honestly. If $i = i^*$ or $i = j^*$, the simulator $\mathfrak{T}$ does as follows:
  - If $i = i^*$, the simulator $\mathfrak{T}$ sets $X_{i^*,k^*} = g^a$ and $H(i^*) = h^c$. For $k \in [\ell]\backslash\{k^*\}$, $\mathfrak{T}$ sets $X_{i^*,k} = g^{x_{i^*,k}}, x_{i^*,k} \leftarrow Z_p^*$. The generation of the remaining signature parts is the same as *Sign* algorithm, except for the generation of these elements $(X_{i^*,1}, \ldots, X_{i^*,\ell})$.
  - If $i = j^*$, the simulator $\mathfrak{T}$ sets $Y_{j^*,k^*} = g^b$. For $k \in [\ell]\backslash\{k^*\}$, $\mathfrak{T}$ generates $Y_{j^*,k}$ as *Sign*.
- *SignChg Queries*: The simulator $\mathfrak{T}$ runs *SignChg* algorithm and returns the signature σ'' to $\mathcal{A}$.
- *Verify Queries*: The simulator $\mathfrak{T}$ runs *Verify* algorithm and returns the result to $\mathcal{A}$.
- *Trace Queries*: The simulator $\mathfrak{T}$ runs *Trace* algorithm and returns the signer's identity to $\mathcal{A}$.

- *Challenge Phase:* The adversary $\mathcal{A}$ adaptively chooses the authorized user's attributes set S^*_{AU} $(P(S^*_{AU}) = 1)$ and the identity aID^*. Then, $\mathcal{A}$ runs *SignChg* algorithm to generate the challenged signature σ^* with the updated data $dm^*(m) = m^* \not\subset am'(m)$. Finally, the adversary $\mathcal{A}$ sends $(S^*_{AU}, m^*, \sigma^*)$ to the simulator $\mathfrak{T}$.

Parse $\sigma^* = (\mu^*, \eta^*, \{\sigma^*_j, X^*_j, Y^*_j\}_{j\in[1,\ell]}, C^*, \sigma^*_{ABS})$. According to the security of EQS, we have $[X^*_1, \ldots, X^*_l]_R = [X_{i',1}, \ldots, X_{i',l}]_R$, and $[Y^*_1, \ldots, Y^*_l]_R = [Y_{j',1}, \ldots, Y_{j',l}]_R$. Thus, $(X^*_1, \ldots, X^*_l) = (X_{i',1}, \ldots, X_{i',l})^r$ and $(Y^*_1, \ldots, Y^*_l) = (Y_{j',1}, \ldots, Y_{j',l})^{rs}$ hold for some $r, s \leftarrow Z_p$.

Suppose $(i', j') = (i^*, j^*)$ and $k' = k^*$, the simulator $\mathfrak{T}$ can get

$$(X^*_{k'})^{\frac{1}{x_{i^*,k'}}} = g^{r\cdot(x_{i^*,k'})\cdot\frac{1}{x_{i^*,k'}}} = g^r$$
$$(Y^*_{k'})^{\frac{1}{(x_{i^*,k'})\cdot(y_{i^*,k'})}} = g^{rs\cdot(x_{i^*,k'})\cdot(y_{i^*,k'})\cdot\frac{1}{(x_{i^*,k'})\cdot(y_{i^*,k'})}} = g^{rs}.$$

Since the adversary $\mathcal{A}$ wins, *Verify* outputs 1. It means that $Y^*_{k^*} = g^{rsb} \neq g, rs \neq 0$. The simulator $\mathfrak{T}$ gets

$$\hat{e}(X^*_{k^*}, \sigma^*_{k^*}) = \hat{e}(Y^*_{k^*}, H(k^*||m^*_{k^*}))$$
$$\hat{e}(X^r_{i^*,k^*}, \sigma^*_{k^*}) = \hat{e}(Y^{rs}_{j^*,k^*}, h^c)$$
$$\hat{e}(g^{ra}, \sigma^*_{k^*}) = \hat{e}(g^{rsb}, h^c)$$
$$\sigma^*_{k^*} = h^{\frac{sbc}{a}}.$$

Finally, the simulator $\mathfrak{T}$ can output $(g^u, g^v, g^x, g^y, h^z) = (g^r, g^{rs}, g^{ra}, g^{rsb}, h^{\frac{sbc}{a}})$ such that

$$\begin{cases} au - x = 0 \\ bv - y = 0 \\ cy - xz = 0 \\ v \neq 0 \end{cases}.$$

It contradicts with Lemma 3. Thus, for a generic group adversary $\mathcal{A}$, who updates the signed message that did not fit the modification description am, it is computationally infeasible to generate a valid signature for the updated message.

Theorem 2 (Transparency). *In the proposed scheme, for an adversary $\mathcal{F}$, it is computationally infeasible to distinguish the signature of updated message from the signature of original message.*

Proof. To prove this theorem, we define a game between a challenger $\mathcal{C}$ and a generic group adversary $\mathcal{F}$. □

Game 2. In the *Game 2*, both the challenger $\mathcal{C}$ and the adversary $\mathcal{F}$ perform as defined in the security definition. That is, the challenger $\mathcal{C}$ runs the *Setup* algorithm and sends the master public key mpk to the adversary $\mathcal{A}$. Then, the adversary $\mathcal{A}$ does as *Query Phase*. Next, the adversary $\mathcal{F}$ adaptively chooses the authorized user's attributes set S^*_{AU} $(P(S^*_{AU}) = 1)$, the identity aID^* and the modification $dm^*(m) = m^* \subset am'(m)$, where the message m is not signed by the data owner or the authorized user. Then, the challenger $\mathcal{C}$ randomly selects $b \leftarrow \{0,1\}$, and sets $m_0 = m, m_1 = m^*$. If $b = 1$, $\mathcal{C}$ runs $SignChg(mpk, pk_{DW}, m, P, \sigma, dm^*(m) = m^* \subseteq am'(m), Ask_{aID,S^*_{AU}})$, and runs $Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am')$ for $b = 0$. Finally, $\mathcal{F}$ returns a guess bit b' for b.

Analysis. If $b = 0$, $\mathcal{C}$ runs $D = Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am')$

$$D = \begin{cases} x_i, y_i \leftarrow Z_p^*, X_i = g^{x_i}, Y_i = X_i^{y_i}, \forall i \in [l] \\ \mu \leftarrow EQS.Sign(sk^*_{EQS}, (X_1, \ldots, X_l)) \\ \eta \leftarrow EQS.Sign(sk^*_{EQS}, (Y_1, \ldots, Y_l)) \\ \sigma_i = H(i||dm(m_i))^{y_i}, \forall i \in [l] \\ \sigma : \begin{cases} \xi_i = \begin{cases} y_i, i \in am \\ 0, otherwise \end{cases} \\ C \leftarrow CP{-}ABE.Enc(mpk, P, am, \{\xi_i\}_{i\in[l]}) \\ \sigma := \{\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^{l}, \mathrm{C}, \sigma_{ABS}\} \end{cases} \end{cases}.$$

If $b = 1$, $\mathcal{C}$ runs $D'' = SignChg(mpk, pk_{DW}, m, P, \sigma, dm^*(m) = m^* \subseteq am'(m), Ask_{aID,S^*_{AU}})$

$$D'' = \begin{cases} r, s \leftarrow Z_p \\ x_i, y_i \leftarrow Z_p^*, X_i = g^{x_i}, Y_i = X_i^{y_i}, \forall i \in [l] \\ \mu' \leftarrow EQS.Sign(sk^*_{EQS}, (X_1, \ldots, X_l)) \\ \eta' \leftarrow EQS.Sign(sk^*_{EQS}, (Y_1, \ldots, Y_l)) \\ \mu = EQS.Chg\mathrm{Rep}(pk_{EQS}, (X_1, \ldots, X_l), \mu', r) \\ \eta = EQS.Chg\mathrm{Rep}(pk_{EQS}, (Y_1, \ldots, Y_l), \eta', rs) \\ \sigma_i = H(i||dm(m_i))^{s\cdot y_i}, \forall i \in [l] \\ \sigma : \begin{cases} \xi_i = \begin{cases} s \cdot y_i, i \in am \\ 0, otherwise \end{cases} \\ C \leftarrow CP{-}ABE.Enc(mpk, P, am, \{\xi_i\}_{i\in[l]}) \\ \sigma := \{\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^{l}, \mathrm{C}, \sigma_{ABS}\} \end{cases} \end{cases}.$$

TABLE 2
Comparison of Functionality Among Our Scheme and Related Schemes

Schemes	Fine-grained Access Control	Controllable Edit	Transparency	Accountability
[5]	×	✓	✓	×
[8]	×	✓	✓	×
[16]	✓	×	×	×
[17]	✓	×	×	×
[21]	✓	×	✓	×
[22]	✓	-	-	-
[23]	✓	×	✓	×
Ours	✓	✓	✓	✓

According to the feature of EQS, the distribution of EQS. Sign is identical to that of EQS.ChgRep. Thus, $D'' = D'$, where

$$D' = \begin{cases} r, s \leftarrow Z_p \\ x_i, y_i \leftarrow Z_p^*, X_i = g^{x_i}, Y_i = X_i^{y_i}, \forall i \in [l] \\ \mu \leftarrow EQS.Sign(sk^*_{EQS}, (X_1, \ldots, X_l)^r) \\ \eta \leftarrow EQS.Sign(sk^*_{EQS}, (Y_1, \ldots, Y_l)^{rs}) \\ \sigma_i = H(i||dm(m_i))^{sy_i}, \forall i \in [l] \\ \sigma : \begin{cases} \xi_i = \begin{cases} s \cdot y_i, i \in am \\ 0, otherwise \end{cases} \\ C \leftarrow CP{-}ABE.Enc(mpk, P, am, \{\xi_i\}_{i \in [l]}) \\ \sigma := \{\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^{l}, \mathrm{C}, \sigma_{ABS}\} \end{cases} \end{cases}.$$

Replacing $r \cdot x_i$ and $s \cdot y_i$ with x_i and y_i, we can get $D' = D$. Furthermore, $D'' = D$. Therefore, we can conclude that the signature of the updated message and the signature of the original message are functionally equivalent. In conclusion, for any PPT adversaries $\mathcal{F}$, the advantage $|\Pr[b' = b] - \frac{1}{2}| < 1/poly(n)$ is negligible for a sufficiently large n, where *poly* stands for a polynomial function.

Theorem 3 (Fine-grained Access Control). *In the proposed scheme, the data owner can develop an access policy without knowing the number and exact identities of the potential authorized users. Only the the potential authorized users who satisfy the access policy can update the portions of the message which are allowed to update.*

Proof. In $Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am)$, the description *am* of admissible modification is encrypted using the CP-ABE on the access policy P. When a potential authorized user wants to update the signed message, he/she should first execute $CP{-}ABE.Decrypt(mpk, C, ABE.Ask_{aID,S_{AU}})$ to get the description *am* of admissible modification. According to the security of CP-ABE, only the potential authorized user whose attribute set satisfies the access policy P can decrypt *am*. Therefore, our proposed scheme achieves fine-grained access control. □

Theorem 4 (Accountability). *In the proposed scheme, TA can extract signer's identity from any valid signature. Thus, the data owner cannot accuse the authorized users (vice versa) of signing.*

Proof. From both algorithms $Sign(mpk, m, P, sk_{DW}, Ask_{dID,S_{DW}}, am)$ and $SignChg(mpk, pk_{DW}, m, P, \sigma, dm, Ask_{aID,S_{AU}})$, we can see that they both contain a traceable attribute-based signature, that are $\sigma_{ABS} \leftarrow TABS.Sign(mpk, sk_{dID,S_{DW}}, m, P)$ and $\sigma'_{ABS} \leftarrow TABS.Sign(mpk, sk_{aID,S_{AU}}, m', P)$. According to the feature of the traceable attribute-based signature, TA can extract signer's exact identity from a valid signature. In conclusion, our scheme supports accountability. □

6 Performance

In this section, we first give functionality comparison among our scheme and several related schemes. Then, we analyze the computational burden of our scheme and the related schemes [9], [16], [21], [23] through several experiments.

6.1 Functionality Comparison

We give functionality comparison among our scheme and the related schemes [5], [8], [16], [17], [21], [22], [23]. As shown in Table 2, our scheme is the only one that satisfies all of the following properties: fine-grained access control, controllable edit, transparency, and accountability. The schemes in [5] and [8] cannot support fine-grained access control. The scheme in [22] did not give the specific construct. Only the scheme in [5], [8] and our scheme can support controllable edit. The scheme in [16] and [17] cannot support transparency. All of these related schemes cannot support accountability.

6.2 Performance Analysis and Comparison

In this section, we first evaluate the performance of our scheme for normal and large-scale files in different scenarios. The normal size of the shared file ranges from 0 to 50MB, and the large scale file ranges from 1 to 100GB. Then, we compare the results with the state-of-the-art schemes in [9], [16], [21], [23] to show the efficiency of our scheme.

The implementation of the proposed data sharing scheme was carried out by using C++ language on a desktop with an Intel Core (TM) i5-4300 CPU @ 2.13 GHz and 8.0 GB RAM. In order to use the existing IT infrastructure of center for mobile cloud computing research (C4MCCR) to perform our experiment, we set up our own Eucalyptus private Infrastructure as a Service (IaaS) cloud. Eucalyptus, an acronym for "Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems", was first proposed to support high performance computing (HPC) research [33]. The implementations of the state-of-the-art schemes in [9], [16], [21], [23] were performed with the help of Pairing-Based

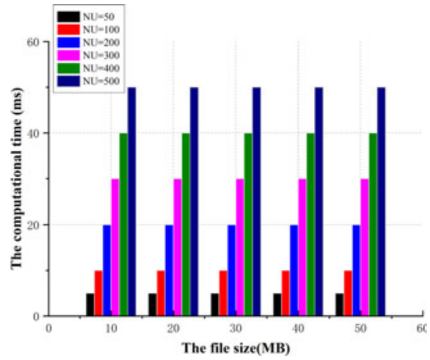

Fig. 4. *KeyGen* phase.

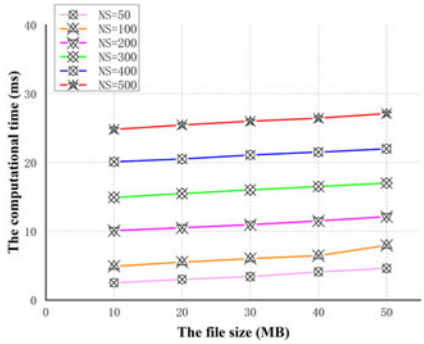

Fig. 5. *Sign* phase.

Cryptography (PBC) version 0.5.14 [34] and the GNU Multiple Precision Arithmetic (GMP) [35]. In the experiments, we use parameter *a.param* in PBC and set the base field size to be 320 KB that is divided into 16,384 blocks, the size of an element in Z_p^* is 20B.

6.2.1 The Computation Cost of the Proposed Scheme With the Normal File Size

We set the size of the file ranges from 0 to 50MB and analyse the computation cost of the proposed data sharing scheme in the *KeyGen*, *Sign*, *SignChg*, and *Verify* phases.

We first evaluate the computational time of *KeyGen* when the number of users ranges from 50 to 500. As shown in Fig. 4, we can see that the computational time of *KeyGen* algorithm is independent of the size of the shared file and is proportional to the number of users. Because the data block size of the same file is different, the number of signatures generated is also different. Thus, we evaluate the computational time of *Sign* when the number of signatures ranges from 50 to 500. As shown in Fig. 5, we can see that the computational time of *Sign* algorithm is proportional to the size of the shared file and increases as the number of signatures increases. In the same way, we can see from Fig. 6 that the computational time of *SignChg* algorithm is proportional to the size of the shared file and increases as the number of signatures increases. Finally, we evaluate the computational time of *Verify* when the number of signatures ranges from 50 to 500. As shown in Fig. 7, we can see that the file size has little effect on the computation time of *Verify* algorithm, while the computation time is proportional to the number of signatures.

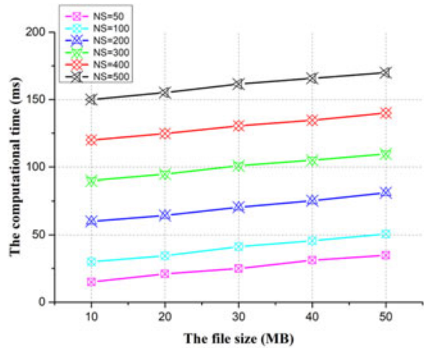

Fig. 6. *SignChg* phase.

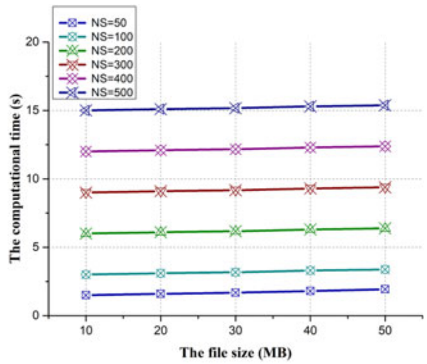

Fig. 7. *Verify* phase.

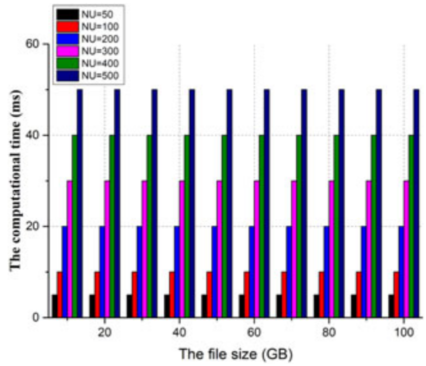

Fig. 8. *KeyGen* phase.

6.2.2 The Computation Cost of the Proposed Scheme With the Large-Scale File Size

We set the size of the file ranges from 10 to 100GB and analyse the computation cost of the proposed data sharing scheme in the *KeyGen*, *Sign*, *SignChg*, and *Verify* phases.

As described above, we also evaluate the computational time of *KeyGen* when the number of users ranges from 50 to 500. As shown in Fig. 8, we can see that the computational time of *KeyGen* algorithm is independent of the size of the shared file and is proportional to the number of users. More specifically, when the size of the shared file is 10GB and the total number of users ranges from 50 to 500, the corresponding computation time varies from 5.04s to 49.96s respectively. Then, we evaluate the computational time of *Sign* when the number of signatures ranges from

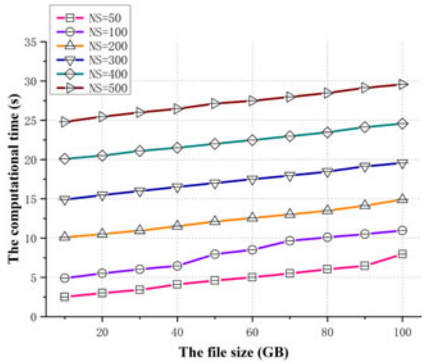

Fig. 9. *Sign* phase.

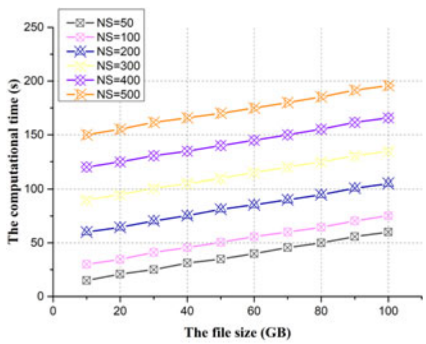

Fig. 10. *SignChg* phase.

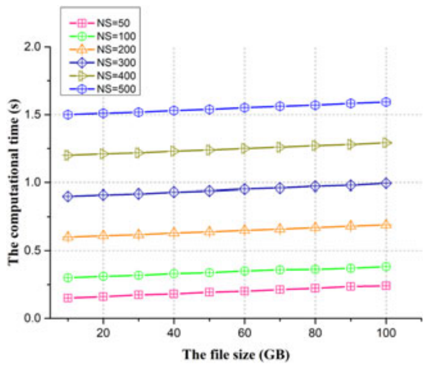

Fig. 11. *Verify* phase.

50 to 500. As shown in Fig. 9, we can see that the computational time of *Sign* algorithm is proportional to the size of the shared file. When the size of the shared file is 10GB and the number of signatures ranges from 50 to 500, the corresponding computation time varies from 2.44s to 24.81s respectively. When the size of the shared file is 100GB and the number of signatures ranges from 50 to 500, the corresponding computation time varies from 7.95s to 29.56s respectively. In the same way, we can see from Fig. 10 that the computational time of *SignChg* algorithm is proportional to the size of the shared file and increases as the number of signatures increases. Finally, we evaluate the computational time of *Verify* when the number of signatures ranges from 50 to 500. As shown in Fig. 11, we can see that the file size has little effect on the computation time of *Verify* algorithm, while the computation time is proportional to the number of signatures.

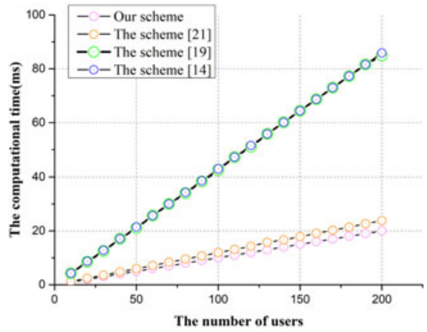

Fig. 12. The computational burden of *KeyGen.*

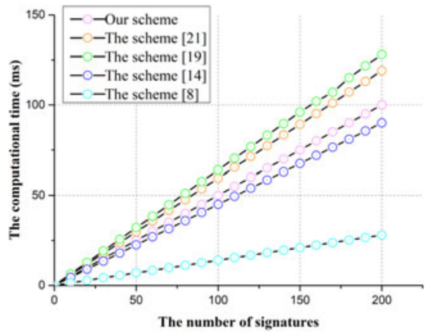

Fig. 13. The computational burden of *Sign.*

6.2.3 The Performance Comparison of the Proposed Scheme With the State-of-the-Art Schemes

In the following experiments, we set the size of the signer's attributes set to be w=4 and the threshold is set to d=3. The monotone span program M has 4 rows and 3 columns. As shown in Fig. 12, we first evaluate the computational time of *KeyGen* when the number of users ranges from 0 to 200. In the schemes [16] and [21], the computation time of *KeyGen* algorithm are linear to the total number of users. Obviously, these two algorithms are the most time consuming. Specifically, when the total number of users ranges from 0 to 200, the corresponding computation time varies from 0ms to 8.59ms. In the scheme [23], the computation time of key generation ranges from 0s to 2.38ms. In the proposed scheme, the computation time of key generation is varies from 0ms to 2ms. Therefore, our scheme is the most efficient.

As shown in Fig. 13, we evaluate the computational time of *Sign* when the number of signatures ranges from 0 to 200. In the most time-consuming schemes [21] and [23], the computation time of *Sign* algorithm are linear to the number of signatures. Specifically, when the number of signatures ranges from 0 to 200, the corresponding computation time varies from 0ms to 12.8ms and 11.9ms respectively. The sanitizable signature scheme [9] is the least time-consuming, while the attribute-based signature scheme [16] is the second least. Our scheme is the most efficient among the attribute-based sanitizable signature schemes and is also more efficient than the naive combination of the attribute-based signature and the sanitizable signature.

As shown in Fig. 14, we evaluate the computational time of *SignChg* when the number of signatures ranges from 0 to 200. In the schemes [21] and [23], the computation time of

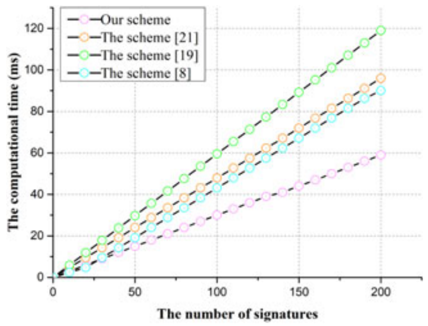

Fig. 14. The computational burden of *SignChg.*

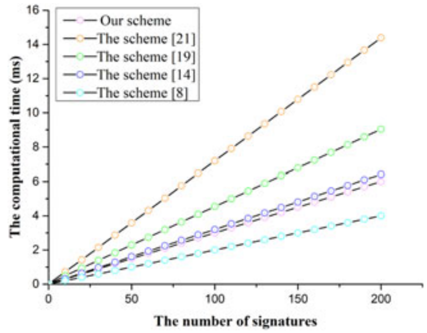

Fig. 15. The computational burden of *Verify.*

SignChg algorithm is linear to the number of signatures. Specifically, when the total number of users ranges from 0 to 200, the corresponding computation time varies from 0ms to 11.9ms and 9.6ms respectively. The sanitizable signature scheme [9] the computation time of *SignChg* ranges from 0ms to 9.04ms. In the proposed scheme, the computation time is varies from 0ms to 5.91ms. Therefore, our scheme is the most efficient.

As shown in Fig. 15, we evaluate the computational time of *Verify* when the number of signatures ranges from 0 to 200. In the schemes [21] and [23], the computation time of *Verify* algorithm are linear to the number of signatures. Obviously, these two algorithms are the most time consuming. Specifically, when the total number of users ranges from 0 to 200, the corresponding computation time varies from 0ms to 9.04s and 14.39s respectively. The sanitizable signature scheme [9] the least time-consuming. Our scheme is the most efficient among the attribute-based sanitizable signature schemes and is also more efficient than the naive combination of the attribute-based signature and the sanitizable signature.

7 Conclusion

In this paper, we proposed a fine-grained and controllably editable data sharing scheme with the malicious user accountability in cloud storage. To ensure the timeliness and the authoritative source of the shared data, the authorized users are allowed to update it on behalf of an authoritative data owner without changing the data source. Only authorized users can update the shared data stored in the external cloud and perform update operations on the portions of the data that are allowed to be updated. The authorized users can convert the signatures of original data into new ones of the updated data without interacting with the data owner. When the incorrect or even harmful information is injected into the shared data, TA can capture and punish the malicious user. Moreover, we designed a new attribute-based sanitizable signature as the underlying technology to support the proposed scheme. The security proof and the experimental analysis demonstrate that the proposed scheme achieves desirable security and efficiency.

The verifier in the proposed scheme needs to compute the time-consuming pairing operation locally, which is included in the *Verify* algorithm. The verifier can be any cloud user, and most verifiers are resource-constrained. Constructing a verifiable outsourced, fine-grained and controllably editable data sharing scheme with accountability for cloud storage is an interesting problem. In our future work, we will focus on designing more sophisticated solutions to the editable data sharing in cloud storage without yielding heavy computational overhead.

Acknowledgments

We are grateful to the anonymous reviewers for their invaluable suggestions.

References

[1] X. Ge *et al.*, "Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 490–504, Jan./Feb. 2021.

[2] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.3011525.

[3] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2005, pp. 159–177.

[4] C. Brzuska *et al.*, "Security of sanitizable signatures revisited," in *Proc. Int. Workshop Public Key Cryptogr.*, 2009, pp. 317–336.

[5] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder, "Unlinkability of sanitizable signatures," in *Proc. Int. Workshop Public Key Cryptogr.*, 2010, pp. 444–461.

[6] S. Canard, F. Laguillaumie, and M. Milhau, "Trapdoor sanitizable signatures and their application to content protection," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2008, pp. 258–276.

[7] J. Lai, X. Ding, and Y. Wu, "Accountable trapdoor sanitizable signatures," in *Proc. Int. Conf. Inf. Secur. Practice Experience*, 2013, pp. 117–131.

[8] K. Miyazaki, G. Hanaoka, and H. Imai, "Digitally signed document sanitizing scheme based on bilinear maps," in *Proc. ACM Symp. Inf. Comput. Commun. Secur.*, 2006, pp. 343–354.

[9] B. Xavier *et al.*, "Efficient invisible and unlinkable sanitizable signatures," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2019, pp. 159–189.

[10] Z. Xu, M. Luo, N. Kumar, P. Vijayakumar, and L. Li, "Privacy-protection scheme based on sanitizable signature for smart mobile medical scenarios," *Wireless Commun. Mobile Comput.*, 2020. [Online]. Available: https://doi.org/10.1155/2020/8877405

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[12] J. Ning, Z. Cao, X. Dong, H. Ma, L. Wei, and K. Liang, "Auditable σ-times outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 94–105, Jan. 2018.

[13] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, and K.-K. R. Choo, "CryptCloud+: Secure and expressive data access control for cloud storage," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 111–124, Jan./Feb. 2021.

[14] H. Cui, R. H. Deng, and G. Wang, "An attribute-based framework for secure communications in vehicular Ad Hoc networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 721–733, Apr. 2019.
[15] N. Eltayieba, R. Elhaboba, A. Hassanb, and F. Li, "A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud," *J. Syst. Archit.*, vol. 102, 2020, Art. no. 101653.
[16] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proc. 5th ACM Symp. Inf. Comput. Commun. Secur.*, 2010, pp. 60–69.
[17] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Proc. Int. Workshop Public Key Cryptogr.*, 2011, pp. 35–52.
[18] J. Su, D. Cao, B. Zhao, X. Wang, and I. You, "ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the internet of things," *Future Gener. Comput. Syst.*, vol. 33, pp. 11–18, 2014.
[19] Y. S. Rao and R. Dutta, "Efficient attribute-based signature and signcryption realizing expressive access structures," *Int. J. Inf. Secur.*, vol. 15, pp. 81–109, 2016.
[20] J. Li, X. Chen, and X. Huang, "New attribute-based authentication and its application in anonymous cloud access service," *Int. J. Web Grid Services*, vol. 11, pp. 125–141, 2015.
[21] X. Liu, J. Ma, J. Xiong, J. Ma, and Q. Li, "Attribute based sanitizable signature scheme," *J. Commun.*, vol. 34, pp. 148–155, 2013.
[22] L. Xu, X. Zhang, X. Wu, and W. Shi, "ABSS: An attribute-based sanitizable signature for integrity of outsourced database with public cloud," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy*, 2015, pp. 167–169.
[23] R. Mo, J. Ma, X. Liu, and Q. Li, "FABSS: Attribute-based sanitizable signature for flexible access structure," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2017, pp. 39–50.
[24] K. Samelin and D. Slamanig, "Policy-Based sanitizable signatures," in *Proc. Cryptographers' Track RSA Conf.*, 2020, pp. 538–563.
[25] C. Hanser and D. Slamanig, "Structure-preserving signatures on equivalence classes and their application to anonymous credentials," *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2014, pp. 491–511.
[26] A. Beimel, "Secret-sharing schemes: A survey," in *Proc. Int. Conf. Coding Cryptol.*, 2011, pp. 11–46.
[27] A. Escala, J. Herranz, and P. Morillo, "Revocable attribute-based signatures with adaptive security in the standard model," in *Proc. 4th Int. Conf. Progress Cryptol. Africa*, 2011, pp. 224–241.
[28] S. Agrawal and M. Chase, "FAME: Fast attribute-based message encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 665–682.
[29] J. Lu, and X. Wang, "Verifiable ring signature," in *Proc. 9th Int. Conf. Distrib. Multimedia Syst.*, 2003, pp. 663–667.
[30] V. Koppula and B. Waters, "Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption," *Adv. Cryptol.*, in A. Boldyreva and D. Micciancio, Eds., Lecture Notes Comput. Sci., vol. 11693, pp. 671–700, 2019, doi: 10.1007/978-3-030-26951-7_23.
[31] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1997, pp. 256–266.
[32] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, 1980.
[33] D. Nurmi *et al.*, "The eucalyptus open-source cloud-computing system," in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2009, pp. 124–131.
[34] Pairing-Based Cryptography (PBC) library. Accessed: Nov. 2020. [Online]. Available: https://crypto.stanford.edu/pbc/howto.html
[35] The GNU Multiple Precision Arithmetic Library (GMP). Accessed: Nov. 2020. [Online]. Available: http://gmplib.org

Huiying Hou received the BS and MS degrees from the College of Computer Science and Technology, Qingdao University, Qingdao, China, in 2015 and 2018, respectively. She is currently working toward the PhD degree in the School of Computer Science and Technology, Fudan University, Shanghai, China. Her research interests include applied cryptography and information security, in particular, vehicle ad-hoc network security and attribute-based cryptology.

Jianting Ning received the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2016. He is currently a professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Computer and Cyber Security, Fujian Normal University, China. Previously, he was a research scientist with the School of Information Systems, Singapore Management University and a research fellow with the Department of Computer Science, National University of Singapore. His research interests include applied cryptography and information security. He has published papers in major conferences/journals such as ACM CCS, ESORICS, ACSAC, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, etc.

Yunlei Zhao received the PhD degree in computer science from Fudan University, Shanghai, China, in 2004. He joined Hewlett-Packard European Research Center, Bristol, U.K., as a post-doctoral researcher, in 2004. Since 2005, he has been with Fudan University, and is currently a distinguished professor with the School of Computer Science, Fudan University. His research interests include the theory and applications of cryptography.

Robert H. Deng (Fellow, IEEE) is AXA chair professor of cybersecurity and director of the Secure Mobile Centre, School of Information Systems, Singapore Management University (SMU). His research interests include the areas of data security and privacy, cloud security, and Internet of Things security. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew Fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards and conference committees. These include the editorial boards of the *IEEE Security & Privacy Magazine*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *Journal of Computer Science and Technology*, and steering committee chair of the ACM Asia Conference on Computer and Communications Security.