Towards Revocable Fine-Grained Encryption of Cloud Data: Reducing Trust upon Cloud

Yanjiang Yang[1], Joseph Liu[2,3], Zhuo Wei[1], and Xinyi Huang[4,5(✉)]

[1] Shield Lab, Huawei Singapore Research Center, Singapore, Singapore
{yang.yanjiang,wei.zhuo}@huawei.com
[2] Faculty of Information Technology, Monash University, Melbourne, Australia
joseph.liu@monash.edu.au
[3] College of Information Engineering, Shenzhen University, Shenzhen, China
[4] School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350108, China
xyhuang@fjnu.edu.cn
[5] State Key Laboratory of Cryptology, Beijing 100878, China

Abstract. ABE (Attribute-based encryption) is capable of fine-grained data encryption, and thus has been studied for secure cloud data sharing. While a number of efforts have been dedicated to resolving the user revocation issue in the multi-user cloud data sharing setting, the trust assumption placed upon the cloud server is still high. In this work, we identify the necessity of achieving *verifiability of cloud decryption* in the proxy-assisted user revocation approach, so as to weaken the trust assumption on the cloud server. We further formulate a model for the system, and present two independent constructions following the formulation. Experimental results show the practicality of our proposed schemes.

Keywords: ABE (Attribute-Based Encryption) · Cloud computing · Fine-grained encryption · User revocation · Authenticated encryption

1 Introduction

Cloud storage services, e.g., Dropbox, Microsoft's Azure storage, and Amazon's S3, provides a wonderful platform for data sharing, enabling users to upload and store their data remotely in the cloud storage as well as to authorize other users to access and download the remotely stored data in real-time [9,11,12]. It is widely recognized that the user data need to be encrypted in order to safeguard against the cloud provider [16,19]. Under this rationale, there have been a number of work proposing to use attribute-based encryption (ABE) [6,14,15,25] to achieve fine-grained access control over cloud data [18,20,27–29,33,34]. Indeed, ABE is a one-to-many public key encryption mechanism in nature, capable of enforcing fine-grained encryption/decryption. In particular, ABE can be categorized into key policy ABE (KP-ABE) and ciphertext policy ABE (CP-ABE). KP-ABE allows data to be encrypted with a set of *attributes*, and each decryption key is

J. Pieprzyk and S. Suriadi (Eds.): ACISP 2017, Part I, LNCS 10342, pp. 127–144, 2017.
DOI: 10.1007/978-3-319-60055-0_7

associated with an *access policy* (defined in terms of attributes); while CP-ABE is complementary – data are encrypted under an access policy, and a decryption key is associated with a set of *attributes*. In either type, a ciphertext can be decrypted using the corresponding decryption key only if the attributes satisfy the access policy.

In the setting of encrypting cloud data with ABE, *user revocation* has been a primary challenge to be resolved. One approach proposed in e.g., [2,24,26,34], is key-update based revocation, where secret key materials are updated to exclude revoked users. This method suffers from poor scalability as all data must be re-encrypted and all remaining legitimate user keys are to be updated or re-distributed, in which case the cost is tremendously high when the data volume or the number of users scales up.

Another approach is to augment ABE schemes with revocation support by incorporating revocation related mechanisms. The ABE schemes in [6,14] propose to include an "expiry time" attribute in the attribute set such that each decryption key is valid only for a limited period of time. The shortcoming of this method is that it does not allow for immediate revocation. In [22], Ostrovsky et. al. propose to include negative constrains in the access policy, such that a revocation of certain attributes amounts to negating these attributes. This mechanism is not scalable in revoking individual users, as each encryption has to involve information of all revoked users each being treated as a distinctive attribute.

More recently, yet another approach was introduced in [29,33] which implements proxy-assisted user revocation, where the cloud server acts as a proxy, and each user's decryption capability is split and represented by two parts, namely: one part is held by the cloud server (i.e., proxy key), and the other part is held by the user. A decryption requires a partial decryption by the cloud server (i.e., cloud decryption or proxy decryption, interchangeably), and a final decryption by the user. For the purpose of user revocation, the cloud server will simply erase the cloud-held proxy key associated with the user to be revoked. This method is particularly promising, as it instantly nullifies a user's decryption privilege while without affecting the legitimate users, requiring no key update or data re-encryption.

While the proxy-assisted user revocation approach demonstrated enormous potential in attaining revocable attribute-based encryption of cloud data, it has been observed in [32] that the constructions in [29,33] are based on a strong assumption – the cloud server is trusted so as not to disclose the revoked users' proxy keys to the revoked users. Considering the possible compromise of the cloud server or the probable existence of the unscrupulous insiders within the cloud service provider, [32] managed to weaken the strong assumption by extending the proxy-assisted with an "all-or-nothing" strategy, such that the cloud server itself is equipped with a public/private key pair and the private key is required in the partial decryption by the cloud server. This means that it is of no use to a revoked user if the cloud server only reveals to the user his/her proxy key, but not the cloud's private key.

Our Contributions. Going along the line of reducing trust upon the cloud server in the proxy-assisted user revocation approach as in [32], we further observe the necessity to provide verifiability of cloud decryption to the authorized users, i.e., to prevent the cloud server from maliciously manipulating encrypted cloud data. Manipulating cloud data by the cloud server could indeed occur, considering the case that some encrypted data records get crashed/lost on the cloud storage, and the cloud server surreptitiously generates bogus records to fool the data owners. We are thus motivated to achieve verifiability of cloud decryption whereby an authorized user is equipped to check the legitimacy of the result of the partial decryption by the cloud server, leading to a further reduction of the trust assumption upon the cloud server. In particular, our contributions are as follows:

- We give a formulation of revocable cloud data sharing with verifiable cloud decryption.
- We present two concrete schemes under the formulation, by extending and progressing the scheme in [32].
- We implement our schemes for experiments, in order to test the practicality of the proposed schemes.

Organization. The remainder of the paper is as follows. Section 2 presents a formulation of the system, followed by two concrete schemes in Sect. 3. Experimental results are given in Sect. 4, and Sect. 5 reviews related works. Section 6 contains concluding remarks.

2 Description and Formulation of the System

2.1 System Setting

As in the proxy-assisted user revocation approach [29,32,33], we consider a cloud storage system consisting of a data owner, a group of data consumers/users, and a cloud server, depicted in Fig. 1. The data owner needs to store its data records at the cloud server, and authorizes the group of users to access the stored data. An example of the entities would be such that the data owner is a company and the data users are the company's employees. Without fully trusting the cloud server, the data owner encrypts its data to ensure the privacy of the data against the cloud server. Data encryption further serves as a measure of built-in fine-grained access control, in such a way that the users have different decryption capabilities based on a pre-defined need-to-know basis. Particular to this system using ABE for data encryption, a user is specified by a set of attributes, e.g., according to the user's functional role in the company, and the user's decryption capability is thus attached to her attributes. The data owner encrypts each data record under an access control policy (specified in terms of attributes), such that a user can successfully decrypt the encrypted record, if and only if the user's attributes satisfy the access policy. As the system works in a multi-user data sharing setting, user revocation is a critical requirement, e.g. when a user leaves

the company. User revocation allows the data owner to revoke a user's ability to decrypt the data (rather than prohibiting the user's access to the encrypted data).

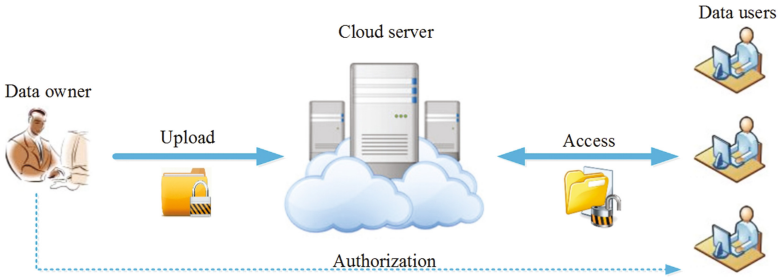

Fig. 1. An overview of the cloud data sharing system

Proxy-assisted User Revocation. To facilitate an understanding of the ensuing formulation of the system, we briefly recall the proxy-assisted user revocation approach [29,32,33]. Specifically, a user's decryption capability is rendered by a proxy key and the user's private key, where the former is held by the cloud server and the latter is possessed by the user. To manage users' proxy keys, the cloud server maintains a list, with each entry containing a user's identity and her corresponding proxy key. When the user requests a data record, the cloud server executes a proxy decryption operation over the data with the user's proxy key (also the cloud's own private key in [32]), generating an intermediate value. The intermediate value is then returned to the user, who gets the plaintext data by a user decryption operation using her private key. As such, to revoke a user it is as simple as to erase the user's proxy key from the cloud server.

2.2 Formulation of the System

As our subsequent constructions will progress the scheme in [32], our formulation extends the model thereof as well by adding in the property of verifiability of cloud/proxy decryption. As such, we first review the formulation in [32], followed by an exposition on the differences incurred due to the addition of verifiability of cloud decryption. Hereafter, familiarity with "attribute" and "access structure" (or "access policy" or "access tree") as introduced in [6,14] is assumed.

Let Λ denote the universe of attributes. A revocable cloud data encryption system specified in [32] comprises the following algorithms:

Setup$(1^\kappa) \rightarrow (params, msk)$**:** Taking as input a security parameter 1^κ, the data owner executes the algorithm to set up public parameters $params$, and a master secret key msk. As usual, $params$ is assumed implicit in the input of the below algorithms.

UKGen$(u) \rightarrow (pk_u, sk_u)$**:** The user key generation algorithm takes as input a user identity, u, and outputs a pair of public/private keys, (pk_u, sk_u), for u. Note that (pk_u, sk_u) is a pair for a standard public key cryptosystem.

PxKGen$(msk, pk_{\mathsf{CS}}, pk_u, \mathbb{A}_u) \rightarrow PxK_u$**:** The proxy key generation algorithm takes as input msk, the cloud server's public key pk_{CS}, a user u's public key pk_u, and the user's attributes, $\mathbb{A}_u \subset \Lambda$, and outputs a proxy key PxK_u for u.

Encrypt$(m, \mathcal{T}) \rightarrow c$**:** The encryption algorithm takes as input a message m, and an access tree $\mathcal{T}$ which specifies an access policy, and outputs a ciphertext c.

PxDec$(sk_{\mathsf{CS}}, PxK_u, c) \rightarrow v$**:** The proxy/cloud decryption algorithm takes as input the cloud server's private key sk_{CS}, a user's proxy key PxK_u, and a ciphertex, c, and outputs an intermediate value v.

UDec$(sk_u, v) \rightarrow m$**:** The user decryption algorithm takes as input a user's private key sk_u, and an intermediate value v, and outputs a plaintext message m.

Revoke$(u, \mathcal{L}_{PxK}) \rightarrow \mathcal{L}'_{PxK}$**:** Taking as input a user identity u, and the Proxy Key list $\mathcal{L}_{PxK}$, the algorithm revokes u's decryption capability by updating and outputting an updated Proxy Key list, $\mathcal{L}'_{PxK}$.

Upon the model, three security requirements, i.e., *Data Privacy Against Cloud Server*, *Data Privacy Against Users* and *User Revocation Support* are specified in [32]. To avoid repetition, we skip the details.

Necessity of Verifiability of Cloud Decryption. In the above formalization, the cloud server is assumed to manage the data owner's data intact and honestly perform the PxDec algorithm. [17,23] studied verifiable outsourced decryption of ABE (where its setting is quite similar to ours if the above PxDec algorithm is understood to be the outsourced decryption of ABE) and identified the necessity of ensuring verifiability of outsourced decryption. We notice that their arguments for the verifiability of outsourced decryption apply to our setting of cloud storage as well, and thus the trust assumption upon the cloud server in the above turns out be a bit strong. This motivates us to investigate providing verifiability to cloud/proxy decryption, reducing the trust assumption on the cloud server.

We further observe that the level of verifiability obtained in [17,23] does not suffice in our setting, and in particular [17,23] did not consider the case where the cloud server who is entrusted for the outsourced decryption to use a bogus but valid ciphertext (in place of the genuine ciphertext) in the outsourced decryption. Note that in [17,23] everyone (including the cloud server) can generate valid ciphertexts (it is public key encryption anyway) and thus the outsourced decryption of these bogus ciphertexts is still valid to the data users; but these ciphertexts are not legitimate, as they are not generated by the data owner. In our setting of cloud data sharing, a higher level of verifiability is desired – it should not only ensure the verifiability as in [17,23], but also enable the data users to verify the legitimacy of the result of cloud decryption. Our idea of formalizing this property is to involve the data owner's master secret key in the Encrypt algorithm.

System Formalization with Verifiability of Cloud Decryption. For brevity, we only highlight the differences that are needed to be imposed to the formulation reviewed above due to the addition of Verifiability of Cloud Decryption. In particular, the changes are restricted to syntax of the Encrypt and UDec algorithms, while other algorithms remain unchanged. Note that in our setting, the verifiability of cloud decryption is checked in the UDec algorithm, rather than the PxDec algorithm.

Encrypt$(msk, m, \mathcal{T}) \rightarrow c$**:** The encryption algorithm takes as input the master secret key msk, message m and an access tree $\mathcal{T}$, and outputs a ciphertext c.

UDec$(sk_u, v) \rightarrow \{m, \perp\}$**:** The user decryption algorithm takes as input a user's private key sk_u, and an intermediate value v, and outputs a plaintext m or $\perp$.

More specifically, the Encrypt algorithm additionally takes as input the master secret key, and the UDec algorithm could output $\perp$ if the verifiability check fails.

Besides the tree security requirements, Data Privacy Against Cloud Server, Data Privacy Against Users and User Revocation Support as defined in [32], one more security requirement, *Verifiability of Cloud Decryption* is to be imposed on the system.

Definition 1 *[Verifiability of Cloud Decryption]. A revocable fine-grained cloud data encryption system satisfies verifiability of cloud decryption if for any PPT adversary, the probability of the adversary winning the following game is* $\epsilon(\kappa)$*, where* $\epsilon(\kappa)$ *is a negligible function with respect to the security parameter* κ*.*

Setup. The challenger runs the Setup algorithm to establish $(params, msk)$, and returns $params$ to the adversary.
Phase 1. The adversary makes a number of data encryption queries, submitting a message m_i and an associated access tree for each query. The challenger executes the $\mathsf{Encrypt}$ algorithm and returns the corresponding ciphertexts to the adversary.
Challenge. The adversary submits an attributes set $\mathbb{A}^*$ and a public key pk_u^* (the corresponding private key is sk_u^*), and the challenger returns the corresponding PxK_u^* generated by executing the PxKGen algorithm.
Phase 2. Phase 1 is repeated.
Output. The adversary outputs a ciphertext c^*. The adversary wins the game if $m = \mathsf{UDec}(sk_u^*, \mathsf{PxDec}(PxK_u^*, c^*)) \neq \perp$ and $m \neq m_i$ for any data encryption query m_i the adversary has asked in Phase 1 and 2.

The formalization essentially captures the requirement that no one (including the cloud server) except the data owner can generate genuine encrypted data records.

3 Our Constructions

As argued above, the level of verifiability obtained in [17,23] does not suffice in our setting, and thus the constructions thereof is not directly applicable to us. In

this section, we present two independent schemes by working upon the scheme in [32] to additionally satisfy Definition 1, giving rise to "authenticated" revocable fine-grained cloud data encryption. The two constructions will take the scheme in [32] as a building block, which is listed in the Appendix for ease of reference.

3.1 Scheme One

In practice, the actual data encryption in [32] would follow the common practice of key encapsulation + data encapsulation (KEM/DEM), namely, an encryption of a data record m is of the form $(\mathsf{Encrypt}(k, \mathcal{T}), \mathsf{SE.Enc}_k(m))$, where SE is a symmetric key block cipher and k is a random key for SE. We also present our two scheme to be working in the mode of KEM/DEM.

This first scheme is inspired by [23] to use randomness extractor as a building block to compensate for the loss of entropy of the data encryption key.

Preliminaries. Let $s \in_R S$ denote an element s randomly chosen from a set S. For a discrete distribution X over $\mathcal{X}$, the min-entropy of X is defined to be $\mathsf{H}_\infty(X) = -\log(\max_{x \in \mathcal{X}} \Pr[X = x])$. The average min-entropy of X conditioned on Y (over $\mathcal{Y}$) is defined as $\tilde{\mathsf{H}}_\infty(X|Y) = -\log E_{y \in \mathcal{Y}}(2^{-\mathsf{H}_\infty(X|Y=y)})$. We recall a lemma in [10] that relates to the security of our scheme: *Let X, Y and Z be random variables. If Y has at most 2^r possible values, then* $\tilde{\mathsf{H}}_\infty(X|(Y,Z)) \geq \tilde{\mathsf{H}}_\infty(X|Z) - r$.

Random Extractor: An efficient function Ext: $\mathcal{X} \times \{0,1\} \rightarrow \mathcal{Y}$ is an average-case (k, ϵ)-strong extractor if for all random variables (X, Z) such that $\tilde{\mathsf{H}}_\infty(X|(Y,Z)) \geq k$, we have $(Z, s, \mathsf{Ext}(X, s)) \approx_\epsilon (Z, s, y \leftarrow_R \mathcal{Y})$, where $s \leftarrow_R \{0,1\}^t$, and $\approx_\epsilon$ denotes the statistical distance upper-bounding by ϵ.

In [10], it is shown that any family of pairwise independent hash functions $\mathcal{H} : \{\hbar : \mathcal{X} \rightarrow \mathcal{Y}\}$ is an average-case $(\tilde{\mathsf{H}}_\infty(X|Z), \epsilon)$-strong extractor if $\tilde{\mathsf{H}}_\infty(X|Z) \geq \log|\mathcal{Y}| + 2\log(1/\epsilon)$.

Construction Details. Following the paradigm of constructing "authenticated" public key encryption, e.g., [1], we reasonably assume the data owner to possess a pair of signing key/verification key (sk, vk) for a digital signature scheme Sig. Under the KEM/DEM paradigm, the encryption of a message m would be $(\mathsf{Encrypt}(msk, k, \mathcal{T}), \mathsf{SE.Enc}_k(m))$. We cannot simply let the data owner sign $\mathsf{SE.Enc}_k(m)$ with sk to provide verifiability of cloud decryption to the data users. This is because the PxDec algorithm may well output a bogus k'. Thus the legitimacy of k must be verified as well to the data users. To this end, we follow the idea of [23]: $H_0(k)$ is published as the verification data for k, where H_0 is a cryptographic hash function. Since $H_0(k)$ reveals at most $|H_0|$ bits of k, a random extractor $\hbar$ is then applied to k to generate a good random key $\tilde{k}$ for SE. Let $\epsilon_\hbar$ is the upper-bound parameter of the extractor $\hbar$, then the parameters must satisfy $0 \leq |\tilde{k}| \leq |k| - |H_0| - 2\log(1/\epsilon_\hbar)$.

We are ready to show how to extend the scheme in [32] (see Appendix), denoted as Basic, to achieve revocable fine-grained cloud data encryption with verifiability of cloud decryption. To avoid repetition, we only show the algorithms to be modified and highlight the extra operations to be added in each of such algorithms.

Setup(1^κ)**:** On input a security parameter 1^κ, the algorithm does the following:

- execute $(params', msk') = \mathsf{Basic.Setup}(1^\kappa)$;
- select a cryptographic hash functions, $H_0 : G_T \rightarrow \{0,1\}^{\ell_0}$, where G_T is contained in $params'$;
- select a semantically secure block cipher $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ with key space $\{0,1\}^{\ell_{\mathsf{SE}}}$;
- select an extractor $\hbar : G_0 \rightarrow \{0,1\}^{\ell_{\mathsf{SE}}}$. Note that let $\epsilon_\hbar$ be the upper-bound parameter of the extractor $\hbar$, then it must satisfy $0 \leq \ell_{\mathsf{SE}} \leq |G_T| - \ell_0 - 2\log(1/\epsilon_\hbar)$[1]
- determine a digital signature scheme $\mathsf{Sig} = (\mathsf{Sig.Sign}, \mathsf{Sig.Verify})$, and select a signing/verification key pair (sk, vk) for Sig;
- set $params = params' \cup \{H_0, \mathsf{SE}, \hbar, \mathsf{Sig}, vk\}$ and $msk = msk' \cup \{sk\}$.

Encrypt$(msk, m, \mathcal{T})$**:** Taking as input the master secret key msk, a message m, and an access tree $\mathcal{T}$, the algorithm works as follows:

- select a random $k \in_R G_T$ and compute $c' = \mathsf{Basic.Encrypt}(k, \mathcal{T})$;
- compute $\tilde{k} = \hbar(k)$, and $\tilde{C} = \mathsf{SE.Enc}_{\tilde{k}}(m)$;
- compute $\sigma = (H_0(k), \mathsf{Sig.Sign}_{sk}(H_0(k)||\tilde{C}))$, where sk is contained in msk;
- set the ciphertext $c = (c', \tilde{C}, \sigma)$.

PxDec$(sk_{\mathsf{CS}}, PxK_u, c = (c', \tilde{C}, \sigma))$**:** The algorithm works as follows:

- Compute $v' = \mathsf{Basic.PxDec}(sk_{\mathsf{CS}}, PxK_u, c')$, and set the intermediate value $v = (v', \tilde{C}, \sigma)$.

UDec$(sk_u, v = (v', \tilde{C}, \sigma))$**:** The algorithm works as follows:

- if $\mathsf{Sig.Verify}_{vk}(\sigma) = 0$, then output $\perp$ and halt;
- compute $k = \mathsf{Basic.UDec}(sk_u, v')$, and test whether $H_0(k)$ equals the corresponding value in σ. If not, then output $\perp$ and halt;
- compute $\tilde{k} = \hbar(k)$ and $m = \mathsf{SE.Dec}_{\tilde{k}}(\tilde{C})$.

Security Analysis. We show that the proposed scheme satisfies the security requirements specified in Sect. 2.

Theorem 1. *The above scheme achieves Data Privacy Against Cloud Server, Data Privacy Against Users, and User Revocation Support, respectively, as specified in* [32].

[1] Let's assume to achieve 80-bit security: G_T could be instantiated such that $|G_T| = 512, \ell_0 = 160, \epsilon_\hbar = 2^{-80}$, then $|G_T| - \ell_0 - 2\log(1/\epsilon_\hbar) = 512 - 160 - 160 = 192$. It is thus more than enough to enable 160-bit block cipher, which can work in an appropriate mode to encrypt message of an arbitrary length.

Intuitively, compared to the scheme [32], the only place that reveals more information in terms of data privacy with respect to both the cloud server and the users is $H_0(k)$. However, this leakage has already been accommodated by the use of the random extractor $\hbar$. Hence data privacy of the scheme is warranted. Formally, we can prove Theorem 1 by a series of hybrid arguments, following the rationale in [23], but we omit the details.

Theorem 2. *The above scheme achieves Verifiability of Cloud Decryption as specified in Definition 1, given that* Sig *is universally unforgeable, and* H_0 *is collisions resistant.*

Proof. Recall that $c^* = (c'^*, \tilde{C}^*, \sigma^* = (H_0^*(k), \mathsf{Sig.Sign}_{sk}(H_0^*(k)||\tilde{C}^*))$ and the corresponding $v^* = (v'^*, \tilde{C}^*, \sigma^*) = \mathsf{PxDec}(sk_{\mathsf{CS}}, PxK_u^*, c^*)$. If the adversary wins the game, then it means that σ^* is a valid signature, but $\mathsf{SE.Dec}_{\hbar(k)}(\tilde{C}^*) \neq m_i$ for any asked query m_i. From this, two cases can be derived:

1. $(\tilde{C}^*, \sigma^*)$ is the reply of one of the data encryption queries the adversary has ever asked, or
2. $(\tilde{C}^*, \sigma^*)$ is not the reply of any data encryption queries the adversary has ever asked.

For case 1, it means that the k^* decrypted from v^* is such that $k^* \neq k$, but $H_0(k^*) = H_0(k)$, in which case a collision of H_0 is found.

For case 2, it means that $(\tilde{C}^*, \sigma^*)$ is a forged signature of the underlying digital signature scheme Sig.

The details are tedious and standard, thus omitted. This completes the proof. □

3.2 Scheme Two

The crust of the first scheme is the explicit protection of the authenticity of the data encryption key k used in the symmetric key cipher, resulting in $H_0(k)$ which leaks information on k; to compensate for the leakage, random extractor $\hbar$ is employed. In this section, we present an alternative scheme which is logically simpler. The rationale is to avoid the explicit protection of k (which has led to $H_0(k)$ and the use of random extractor $\hbar$ in the first scheme); instead, authenticated encryption (e.g., [5,7]) keyed by k is used for data encryption, replacing the symmetric key cipher used in the first scheme.

A Review of Authenticated Encryption. Authenticated encryption [5,7] is a well-established symmetric key cryptosystem, simultaneously providing confidentiality and integrity protection of the encrypted messages. Specifically, an authenticated encryption scheme is $\mathsf{AE} = (\mathsf{AE.Enc}, \mathsf{AE.Dec})$, where Enc and Dec are encryption algorithm and decryption algorithm, respectively. Compared to symmetric key block cipher, authenticated encryption not only protects the confidentiality/privacy of the messages, but also the integrity of the messages. Concretely, $\mathsf{AE.Dec}$ could output $\perp$ if the ciphertext to be decrypted is invalid. This

property is often naturally formalized as *integrity of ciphertexts*, i.e., AE.Dec checks the integrity of a ciphertext and gives up if the ciphertext is not legitimate/authenticated.

The formalization of confidentiality/privacy protection of authenticated encryption is identical to block cipher, so we do not repeat. For a better understanding of integrity protection, below is a formulation of integrity protection of ciphertexts of authenticated encryption.

1. The challenger chooses a key k for AE.
2. The adversary makes a number of encryption queries, submitting messages $m_1, m_2, \cdots$. For each query m_i, the challenger computes $c_i = \mathsf{AE.Enc}_k(m_i)$ and returns c_i to the adversary.
3. Finally, the adversary outputs a ciphertext c^*. The adversary wins if $\mathsf{AE.Dec}_k(c^*) \neq \perp$ and c^* is not any c_i returned by the challenger in step 2.

Integrity protection of ciphertexts stipulates that the probability of the adversary wins is negligible.

NOTE. We point out that implicit in this formulation is that the adversary cannot make a valid ciphertext under a key k still valid under a different key k'. To see this, in a strive to output a valid v^* under k, the adversary itself can anyway generate a different k' and in turn generate as many valid ciphertexts under k' as it wishes to. So an alternative formalism would be such that at the end, the adversary outputs a key $k^* \neq k$ and a ciphertext $c^* \in \{c_1, c_2, \cdots\}$, and the adversary wins if $\mathsf{AE.Dec}_{k^*}(c^*) \neq \perp$.

Construction Details. Based on the above discussions, encryption of a message m is of the form $(\mathsf{Encrypt}(msk, k, \mathcal{T}), \mathsf{AE.Enc}_k(m))$, where AE is an authenticated encryption scheme. This time, the data owner can simply sign $\mathsf{AE.Enc}_k(m)$ to attain verification of cloud decryption. Since k is directly used to key-up AE, there is no extra leakage of k.

The presentation of the scheme is still in the form of extending Basic, as in the earlier first scheme.

Setup(1^κ)**:** The algorithm does the following:

- execute $(params', msk') = \mathsf{Basic.Setup}(1^\kappa)$;
- select an authenticated encryption scheme AE = (AE.Enc, AE.Dec) with key space $\{0,1\}^{\ell_{\mathsf{AE}}}$;
- determine a digital signature scheme Sig = (Sig.Sign, Sig.Verify), and select a signing/verification key pair (sk, vk) for Sig;
- set $params = params' \cup \{\mathsf{AE}, \mathsf{Sig}, vk\}$ and $msk = msk' \cup \{sk\}$.

Encrypt$(msk, m, \mathcal{T})$**:** The algorithm works as follows:

- select a random $k \in_R \{0,1\}^{\ell_{\mathsf{AE}}}$ and compute $c' = \mathsf{Basic.Encrypt}(k, \mathcal{T})$;
- compute $\tilde{C} = \mathsf{AE.Enc}_k(m)$;
- compute $\sigma = \mathsf{Sig.Sign}_{sk}(\tilde{C})$;

- set the ciphertext $c = (c', \tilde{C}, \sigma)$.

PxDec$(sk_{\mathsf{CS}}, PxK_u, c = (c', \tilde{C}, \sigma))$**:** The algorithm works as follows:

- Compute $v' = \mathsf{Basic.PxDec}(sk_{\mathsf{CS}}, PxK_u, c')$, and set the intermediate value $v = (v', \tilde{C}, \sigma)$.

UDec$(sk_u, v = (v', \tilde{C}, \sigma))$**:** The algorithm works as follows:

- if $\mathsf{Sig.Verify}_{vk}(\sigma) = 0$, then output $\perp$ and halt;
- compute $k = \mathsf{Basic.UDec}(sk_u, v')$;
- compute $m = \mathsf{AE.Dec}_k(\tilde{C})$.

Security Analysis. Since authenticated encryption and block cipher is the same in terms of data privacy protection, we only show that this scheme achieves verifiability of cloud decryption.

Theorem 3. *The scheme shown above achieves Verifiability of Cloud Decryption as specified in Definition 1, given that* Sig *is universally unforgeable and AE satisfies integrity protection of ciphertexts.*

Proof. Recall that $c^* = (c'^*, \tilde{C}^*, \sigma^* = \mathsf{Sig.Sign}_{sk}(\tilde{C}^*))$. If the adversary wins the game, then it means that σ^* is a valid signature upon $\tilde{C}^*$, but $\mathsf{SE.Dec}_k(\tilde{C}^*) \neq m_i$ for any asked query m_i, where $k = \mathsf{Basic.UDec}(sk_u^*, c'^*)$. From this, two cases can be derived:

1. $(\tilde{C}^*, \sigma^*)$ is the reply of one of the data encryption queries the adversary has ever asked, or
2. $(\tilde{C}^*, \sigma^*)$ is not the reply of any data encryption queries the adversary has ever asked.

For case 1, it means that $k = \mathsf{Basic.UDec}(sk_u^*, c'^*)$ is not the same as the original key used to generate $\tilde{C}^*$. This directly contradicts integrity of ciphertexts of authenticated encryption (see the alternative formalism discussed earlier).

For case 2, it means that $(\tilde{C}^*, \sigma^*)$ is a forged signature of the underlying digital signature scheme Sig.

Due to limited space, we omit the details of the proof which will be provided in the full version of this paper. This completes the proof. □

4 Experimental Results

To evaluate the performance of our proposed schemes, we did extensive experiments. The implementation is based on the Pairing-Based Cryptography (PBC) library (https://crypto.stanford.edu/pbc/). The bilinear map $e : G_0 \times G_0 \rightarrow G_T$ in our schemes is instantiated with a 512-bit supersingular curve of embedding degree, with $|p| = 160$ (p is the prime order of G_0 and G_T). Other cryptographic primitives used include RSA digital signature, AES for block cipher, AES-GCM for authenticated encryption. In addition, SHA256 is used in place of a random extractor.

Experimental Results. In practical cloud storage services, the performance at the cloud server's side and at the user's side is of concern, which directly relates to the PxDec algorithm and the UDec algorithm in our schemes. Our experiments thus mainly gauge the computational performance of these two algorithms in our schemes. Since our schemes are built upon Basic, the scheme in [32], we also implemented Basic as the baseline for comparison.

Performance of Proxy Decryption. We run the PxDec algorithm on a desktop PC with 2.66 GHz Intel Core2Duo and 3.25 GB RAM. The PxDec algorithm is fed a set of all-AND access trees, i.e., an access tree with all non-leaf being "AND" gates. The reason is that an access policy in the form of all-AND tree is expected to impose the heaviest workload in the PxDec algorithm, compared to the access tree with the same number of leaf nodes. The experimental results are shown in Fig. 2, which demonstrates timing (i.e., computational performance of the three schemes) with respect to the number of attributes (leaf nodes). The experimental results are the average of repeating each experiment for 100 times.

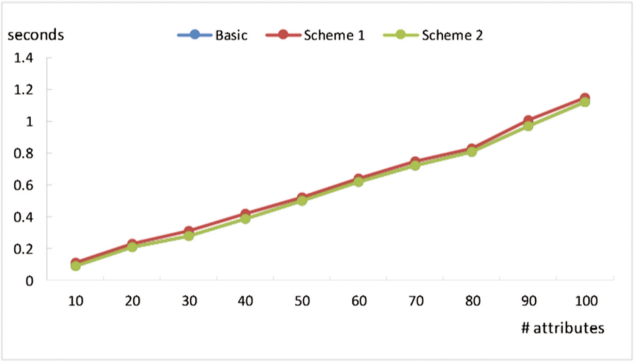

Fig. 2. Computational performance of PxDec of three schemes

As evident from the Figure, (1) the three schemes have identical performance in proxy decryption. This is apparent from the construction of our two schemes; (2) the experimental results show that the PxDec algorithm performs linear computations with respect to the number of attributes; (3) it takes about 1.2 s to perform the PxDec algorithm in case of 100 attributes. Such a performance should be acceptable for practical applications.

Performance of User Decryption. We run the UDec algorithm of the three schemes on a smartphone configured with a 1.2 GHz CPU and 2 GB RAM. The experimental results are depicted in Fig. 3, which indicates that on average, it takes about 50 ms to decrypt a ciphertext in the Basic scheme, and about 110 ms by our two schemes. The extra time taken in our schemes is mainly for digital signature verification. These results suggest that it is indeed affordable for a resource constraint device to perform the UDec algorithm.

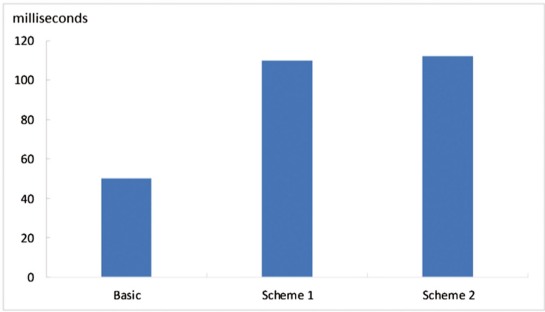

Fig. 3. Computational performance of user decryption

5 Related Work

Cloud Data Encryption with ABE. A large number of cloud data encryption schemes have been proposed in the literature. Of particular relevance to us are those utilizing ABE. As an one-to-many encryption scheme, ABE is required to provide user revocation support if deployed for encryption of cloud data.

Yu et al. [34] suggested adopting KP-ABE to achieve fine-grained data sharing. To support user revocation, they proposed using the proxy re-encryption (PRE) technique [3] to update users' decryption keys. In this approach, the bulk of the computationally expensive operations (e.g. re-generation of encrypted cloud data due to user revocation) are performed by the cloud server. Although a cloud server generally has significantly more computational resources, each user's quota is cost based. Similar limitation is observed in the scheme proposed by Wang et al. [26]. Sahai et al. [24] proposed an attribute revocable CP-ABE scheme, using ciphertext delegation and the piecewise property of private keys. In particular, the system proceeds in epochs, and in each epoch, the attribute authority generates a set of update keys (as the other piece of each private key) according to the revocation list. All the ciphertexts are then re-encrypted with a new access policy (the principal access policy remains unchanged, but the extra access policy changes in each epoch). A similar attribute revocation method has also been explored in the multi-authority setting [30,31], where users' attributes are issued by multiple independent attribute authorities. Similar to other ABE schemes with built-in attribute revocation support (such as expiry time and negative attributes), these schemes face the challenge of transforming attribute revocation into efficient revocation for individual users. For example, the limitation in the scheme proposed by Liu et al. [21] that uses the "expiry time" mechanism for user revocation is the inability to support real-time or immediate revocation. To sum up, the overhead introduced by these schemes in the re-generation of encrypted data and key update is large, although some have managed to push much of the overhead for the cloud server to perform.

Yang et al. [29] were the first to propose proxy-assisted user revocation in using ABE for secure cloud data sharing. The proxy-assisted user revocation

approach actually implements decryption capability splitting, where a data user's complete decryption capability is split into two parts – one is taken on by the cloud server as a proxy while the other is taken on by the user herself. Subsequent work such as [32,33] improved over [29] by reducing the strong trust assumption upon the cloud server. Our work in this paper goes along this same line of research, progressing [29,32,33] by further weakening the trust assumption.

We point out that "decryption capability splitting" contrasts with "decryption key splitting" to be reviewed shortly, and the two differ mainly in the way keys are generated: in the latter, the key shares held by the proxy and the user are generated by a single trusted entity; as a result, it suffers from the issue of key escrow, from the user's point of view. In contrast, "decryption capability splitting" does not have the key escrow problem, as a user can generate her own key and does need not disclose it to others. It is expected that "decryption capability splitting" would be advantageous over "decryption key splitting" in many applications.

Key-Split Cryptography. Boneh et al. [4] proposed "mediated RSA" to split the private key of RSA into two shares, such that one share is delegated to an online "mediator" (mediator is a similar concept as proxy in our setting) and the other is given to the user. As RSA decryption and signing require the collaboration of both parties, the user's cryptographic capabilities are immediately revoked if the mediator does not cooperate. Recently, Chen et al. [8] presented a mediated CP-ABE scheme, where the mediator's key is issued over a set of attributes. The scheme in [13] and the follow-up work [17,23] can also be viewed as mediated ABE, although the purpose of these work is to outsource the costly ABE decryption to the mediator, instead of for immediate revocation. As a final note, our work in this paper is inspired by [17,23] to provide verifiability of cloud decryption, but we end up desiring and attaining a higher level of verifiability.

6 Conclusions

In this paper, we went further along the line of reducing trust assumption upon the cloud server in the proxy-assisted user revocation approach. In particular, we first identified the necessity of achieving *verifiability of cloud decryption*, and then gave a formulation of the system; then two concrete schemes were presented; experiments were conducted and promising experimental results were obtained. Our work in this paper walked revocable fine-grained cloud data sharing a step further towards practical deployment.

Acknowledgments. Joseph K. Liu is supported by the Science and Technology Innovation Projects of Shenzhen (GJHZ20160226202520268). Xinyi Huang is supported by the Distinguished Young Scholars Fund of Fujian (2016J06013) and the State Key Laboratory of Cryptology Research Fund.

Appendix: A Review of the Scheme in [32]

The details of the scheme in [32] are as follows.

Setup(1^κ)**:** On input a security parameter 1^κ, the algorithm:

- determines a bilinear map, $e : G_0 \times G_0 \rightarrow G_T$, where G_0 and G_T are cyclic groups of κ-bit prime order p;
- selects g, which is a generator of G_0;
- selects a cryptographic hash function, $H : \{0,1\}^* \rightarrow G_0$;
- picks $\alpha, \beta \in_R Z_p$, and sets $params = (e, G_0, g, h = g^\beta, \mathcal{G}_\alpha = e(g,g)^\alpha)$ and $msk = (\alpha, \beta)$.

UKGen(u)**:** On input a user identity u, the algorithm chooses $x_u \in_R Z_p$, and sets $(pk_u = g^{x_u}, sk_u = x_u)$. It can be seen that (pk_u, sk_u) is a standard ElGamal type key pair. The cloud server also uses this algorithm to generate a key pair, $(pk_{\mathsf{CS}} = g^{x_{\mathsf{CS}}}, sk_{\mathsf{CS}} = x_{\mathsf{CS}})$.

PxKGen$(msk, pk_{\mathsf{CS}}, pk_u, \mathbb{A}_u)$**:** On input $msk = (\alpha, \beta), pk_{\mathsf{CS}} = g^{x_{\mathsf{CS}}}, pk_u = g^{x_u}$ and $\mathbb{A}_u$, the algorithm chooses $r_1, r_2, r_i \in_R Z_p, \forall i \in \mathbb{A}_u$, and sets

$$PxK_u = (k = (pk_{\mathsf{CS}}^{r_1} pk_u^\alpha g^{r_2})^{\frac{1}{\beta}}, k' = g^{r_1}, \forall i \in \mathbb{A}_u : \{k_{i1} = g^{r_2} H(i)^{r_i}, k_{i2} = g^{r_i}\})$$

Encrypt$(m, \mathcal{T})$**:** Taking as input a message, m, and $\mathcal{T}$, the algorithm works as follows: Firstly, it selects a polynomial, q_n, for each node, n, (including the leaf nodes) in $\mathcal{T}$. These polynomials are chosen in a top-down manner starting from the root node, rt. For each node n, set the degree d_n of the polynomial q_n to be $d_n = t_n - 1$, where t_n is the threshold value of node n. Starting with the root node, rt, the algorithm chooses an $s \in_R Z_p$, and sets $q_{rt}(0) = s$. It next selects d_{rt} other random points to define q_{rt} completely. For any other node n, it sets $q_n(0) = q_{\mathsf{parent}(n)}(\mathsf{index}(n))$, and chooses d_n other points to define q_n. Let L be the set of leaf nodes in $\mathcal{T}$. The algorithm sets the ciphertext, c, as

$$c = (\mathcal{T}, C = m \cdot \mathcal{G}_\alpha^s, C' = h^s, C'' = g^s, \\ \forall \ell \in L : \{C_{\ell 1} = g^{q_l(0)}, C_{\ell 2} = H(\mathsf{att}(\ell))^{q_l(0)}\})$$

PxDec$(sk_{\mathsf{CS}}, PxK_u, c)$**:** On input $sk_{\mathsf{CS}} = x_{\mathsf{CS}}$, and $PxK_u = (k, k', \forall i \in \mathbb{A}_u : \{k_{i1}, k_{i2}\})$ associating with a set of attributes, $\mathbb{A}_u$, and a ciphertext, $c = (\mathcal{T}, C, C', C'', \forall \ell \in L : \{C_{\ell 1}, C_{\ell 2}\})$, the algorithm outputs an intermediate value, v if $\mathcal{T}(\mathbb{A}_u) = 1$, and $\perp$ otherwise. Specifically, the algorithm is recursive. We first define an algorithm, $\mathsf{DecNd}_n(PxK_u, c)$, on a node, n, of $\mathcal{T}$. If node, n, is a leaf node, we let $z = \mathsf{att}(n)$ and define as follows: $z \notin \mathbb{A}_u$, $\mathsf{DecNd}_n(PxK_u, c) = \perp$; otherwise $\mathsf{DecNd}_n(PxK_u, c) = F_n$, where

$$F_n = \frac{e(k_{z1}, C_{n1})}{e(k_{z2}, C_{n2})} = \frac{e(g^{r_2} H(z)^{r_z}, g^{q_n(0)})}{e(g^{r_z}, H(z)^{q_n(0)})} = e(g,g)^{r_2 \cdot q_n(0)} \tag{1}$$

We now consider the recursive case when n is a non-leaf node. The algorithm, $\mathsf{DecNd}_n(PxK_u, c)$, then works as follows. For each child node ch of n, it calls $\mathsf{DecNd}_{ch}(PxK_u, c)$, and stores the output as F_{ch}. Let S_n be an arbitrary t_n-sized set of child nodes, ch, such that $F_{ch} \neq \perp$. If such a set does not exist, then the node is not satisfied and $\mathsf{DecNd}_n(PxK_u, c) = F_n = \perp$. Otherwise,

we let the Lagrange coefficient, $\triangle_{i,S}$ for $i \in Z_p$, and a set S of elements in Z_p be $\triangle_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We next compute

$$\begin{aligned} F_n &= \prod_{ch \in S_n} F_{ch}^{\triangle_{i,S'_n}(0)}, \text{where } \begin{smallmatrix} i=\mathsf{index}(ch), \\ S'_n=\{\mathsf{index}(ch): ch \in S_n\} \end{smallmatrix} \\ &= \prod_{ch \in S_n} (e(g,g)^{r_2 \cdot q_{ch}(0)})^{\triangle_{i,S'_n}(0)} \\ &= \prod_{ch \in S_n} (e(g,g)^{r_2 \cdot q_{\mathrm{parent}(ch)}(\mathrm{index}(ch))})^{\triangle_{i,S'_n}(0)} \\ &= \prod_{ch \in S_n} (e(g,g)^{r_2 \cdot q_n(i)})^{\triangle_{i,S'_n}(0)} \\ &= e(g,g)^{r_2 \cdot q_n(0)} \end{aligned} \tag{2}$$

In this way, $\mathsf{DecNd}_{rt}(PxK_u, c)$ for the root node rt can be computed if $\mathcal{T}_{rt}(\mathbb{A}_u) = 1$, where $\mathsf{DecNd}_{rt}(PxK_u, c) = e(g,g)^{r_2 \cdot q_{rt}(0)} = e(g,g)^{r_2 \cdot s} = F_{rt}$. Next, the proxy decryption algorithm computes

$$\frac{e(k, C')}{e(k', C'')^{x_{\mathsf{CS}}} F_{rt}} = \frac{e((pk_{\mathsf{CS}}^{r_1} pk_u^{\alpha} g^{r_2})^{\frac{1}{\beta}}, h^s)}{e(g^{r_1}, g^s)^{x_{\mathsf{CS}}} e(g,g)^{r_2 \cdot s}} = e(pk_u, g)^{s \cdot \alpha}.$$

Finally, it sets $v = (C = m \cdot \mathcal{G}_{\alpha}^s, e(pk_u, g)^{s \cdot \alpha})$.

UDec(sk_u, v)**:** On input a user private key, $sk_u = x_u$, and an intermediate value, $v = (C = m \cdot \mathcal{G}_{\alpha}^s, e(pk_u, g)^{s \cdot \alpha})$, the user decryption algorithm computes $\frac{m \cdot \mathcal{G}_{\alpha}^s}{(e(pk_u, g)^{s \cdot \alpha})^{x_u^{-1}}} = m$.

Revoke$(u, \mathcal{L}_{PxK})$**:** On input a user identity, u, and the Proxy Key list, $\mathcal{L}_{PxK}$, the user revoking algorithm deletes the entry corresponding to u from the list – i.e. $\mathcal{L}'_{PxK} = \mathcal{L}_{PxK} \backslash \{u, PxK_u\}$. In a real world application, an interface should be provided to the data owner for the data owner to perform the update in real-time.

References

1. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_6
2. Attrapadung, N., Imai, H.: Attribute-based encryption supporting direct/indirect revocation modes. In: Proceedings of the IMA International Conference on Cryptography and Coding, pp. 278–300 (2009)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). doi:10.1007/BFb0054122
4. Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. In: Proceedings of the USENIX Security (2001)

5. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000). doi:10.1007/3-540-44448-3_41
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of IEEE S&P (2007)
7. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html
8. Chen, Y., Jiang, L., Yiu, S.M., Au, M., Xuan, W.: Fully-RCCA-CCA-Secure ciphertext-policy attribute based encryption with security mediator. In: Proceedings of the 16th International Conference on Information and Communications Security, ICICS 2014 (2014)
9. Cloud Security Alliance: Security guidance for critical areas of focus in cloud computing (2009). http://www.cloudsecurityalliance.org
10. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractor: how to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008)
11. Network, E., Agency, I.S.: Cloud computing risk assessment. http://www.enisa.europa.eu/act/rm/_les/deliverables/cloud-computing-risk-assessment
12. Gartner: Don't trust cloud provider to protect your corporate assets, 28 May 2012. http://www.mis-asia.com/resource/cloud-computing/gartner-dont-trust-cloud-provider-to-protect-your-corporate-assets
13. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: Proceedings of the USENIX Security (2011)
14. Goyal, V., Pandy, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the ACM CCS 2006 (2006)
15. Hohenberger, S., Waters, B.: Online/Offline attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 293–310. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_17
16. Jiang, T., Chen, X., Li, J., Wong, D.S., Ma, J., Liu, J.K.: Towards secure and reliable cloud storage against data re-outsourcing. Future Gener. Comp. Syst. **52**, 86–94 (2015)
17. Lai, J., Deng, R.H., Guan, C., Weng, J.: Attribute-based encryption with verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **8**(8), 1343–1354 (2013)
18. Liang, K., Au, M.H., Liu, J.K., Susilo, W., Wong, D.S., Yang, G., Yu, Y., Yang, A.: A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing. Future Gener. Comp. Syst. **52**, 95–108 (2015)
19. Liang, K., Susilo, W., Liu, J.K.: Privacy-preserving ciphertext multi-sharing control for big data storage. IEEE Trans. Inf. Forensics Secur. **10**(8), 1578–1589 (2015)
20. Liu, Z., Wong, D.S.: Practical attribute based encryption: traitor tracing, revocation, and large universe. https://eprint.iacr.org/2014/616.pdf
21. Liu, J., Wan, Z., Gu, M.: Hierarchical attribute-set based encryption for scalable, flexible and fine-grained access control in cloud computing. In: Proceedings of the 7th Information Security Practice and Experience Conference, ISPEC 2011 (2011)
22. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proceedings of ACM CCS 2007, pp. 195–203 (2007)
23. Qin, B., Deng, R.H., Liu, S., Ma, S.: Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **10**(7), 1384–1393 (2015)

24. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Proceedings of Advances in Cryptology, Crypto 2012, pp. 199–217 (2012)
25. Waters, B.: Ciphertext-policy attribute-Based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of Practice and Theory in Public Key Cryptography, PKC 2011, pp. 53–70 (2011)
26. Wang, G., Liu, Q., Wu, J.: Hierarhical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of ACM CCS 2010 (2010)
27. Wang, S., Zhou, J., Liu, J.K., Yu, J., Chen, J., Xie, W.: An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE Trans. Inf. Forensics Secur. **11**(6), 1265–1277 (2016)
28. Wang, S., Liang, K., Liu, J.K., Chen, J., Yu, J., Xie, W.: Attribute-based data sharing scheme revisited in cloud computing. IEEE Trans. Inf. Forensics Secur. **11**(8), 1661–1673 (2016)
29. Yang, Y., Ding, X., Lu, H., Wan, Z., Zhou, J.: Achieving revocable fine-grained cryptographic access control over cloud data. In: Proceedings of the 16th Information Security Conference, ISC 2013 (2013)
30. Yang, K., Jia, X.: Expressive, efficient, and revocable data access control for multi-authority cloud storage. IEEE Trans. Parallel Distrib. Syst. **25**(7), 1735–1744 (2014)
31. Yang, K., Jia, X., Ren, K., Zhang, B., Xie, R.: DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems. IEEE Trans. Inf. Forensics Secur. **8**(11), 1790–1801 (2013)
32. Yang, Y., Liu, J.K., Liang, K., Choo, K.-K.R., Zhou, J.: Extended proxy-assisted approach: achieving revocable fine-grained encryption of cloud data. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 146–166. Springer, Cham (2015). doi:10.1007/978-3-319-24177-7_8
33. Yang, Y., Lu, H., Weng, J., Zhang, Y., Sakurai, K.: Fine-grained conditional proxy re-encryption and application. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) ProvSec 2014. LNCS, vol. 8782, pp. 206–222. Springer, Cham (2014). doi:10.1007/978-3-319-12475-9_15
34. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of IEEE INFOCOM 2010 (2010)