01 Mar 2020

A Collusion-Resistant Revocable Attribute-Based Encryption Scheme for Secure Data Sharing in Cloud

Azharul Islam

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

A collusion-resistant revocable attribute-based encryption scheme for secure data sharing in cloud

1st Md Azharul Islam
dept. of Computer Science
Missouri University of Science & Technology
Rolla, MO, USA.
mdazharul.islam@mst.edu

2nd Sanjay Madria
dept. of Computer Science
Missouri University of Science & Technology
Rolla, MO, USA.
madrias@mst.edu

***Abstract*—Attribute-based encryption (ABE) is a prominent cryptographic tool for secure data sharing in the cloud because it can be used to enforce very expressive and fine-grained access control on outsourced data. The revocation in ABE remains a challenging problem as most of the revocation techniques available today, suffer from the collusion attack. The revocable ABE schemes which are collusion resistant require the aid of a semi-trusted manager to achieve revocation. More specifically, the semi-trusted manager needs to update the secret keys of nonrevoked users followed by a revocation. This introduces computation and communication overhead, and also increases the overall security vulnerability. In this work, we propose a revocable ABE scheme that is collusion resistant and does not require any semi-trusted entity. In our scheme, the secret keys of the nonrevoked users are never affected. Our decryption requires only an additional pairing operation compared to the baseline ABE scheme. We are able to achieve these at the cost of a little increase (compared to the baseline scheme) in the size of the secret key and the ciphertext. Theoretical performance analysis and experimental results show that our scheme outperforms the relatable existing schemes.**

***Index Terms*—secure cloud data sharing, attribute based encryption, revocation**

I. Introduction

Many companies and organizations often outsource their data to a public cloud to enjoy advantages such as availability, scalability, and lower maintenance cost offered by the cloud. However, confidentiality and access control of the outsourced data remains a concern since the data owner loses control over the data once it is uploaded to the cloud. Recently, attribute-based encryption (ABE) has become a very promising tool [1] to achieve confidentiality and fine-grained access control for data outsourcing in the cloud. ABE allows a data owner to encrypt his or her data using a policy expressed in terms of a set of attributes so that it can be decrypted only if the secret key has enough attributes to satisfy the policy. Let us consider the motivational example in the following section.

A. A motivational example

Suppose CryptoFlix is a Netflix-like streaming service that keeps all its media contents in a public cloud run by a third-party cloud service provider. Before outsourcing media files to the public cloud, it encrypts them using attribute-based encryption scheme. CryptoFlix offers two types of subscription plans: basic and premium. Basic and premium plans allow a subscriber to choose three and five attributes, respectively. Let us assume that two users, Alice and Bob, have a subscription for the basic plan, and another user, Eve has a subscription for the premium plan. Alice loves science fiction movies and documentaries, she gets decryption keys for attributes *movie, scifi,* and *documentary* from the attribute authority (a trusted entity responsible for generating and distributing attribute secret keys). Conversely, Bob loves newly released tv shows and gets decryption keys for *new_release, tv_show,* and *documentary* attributes from attribute authority. Let Eve has decryption keys for attribute *movie, tv_show, documentary, scifi,* and *new_release*, but her premium subscription plan recently expired. CryptoFlix encrypts the newly released science fiction movie XFiction under the policy *(new_release AND movie AND scifi)* and uploads it to a public cloud. Note that none of the three users can decrypt XFiction under normal circumstance. However, they can cooperate with each other and try to decrypt it by launching the following attacks:

- *Type I attack:* Multiple users who individually do not have enough attributes to satisfy a policy cooperate with each other so that their collective attribute keys may satisfy the policy.
- *Type II attack:* A revoked user who cannot decrypt a file despite having enough attributes to satisfy the policy cooperates with a nonrevoked user to restore his or her decryption ability in order to decrypt the file.

These attacks are called collusion attack. In a *type I* attack, Alice and Bob would combine their attribute keys and try to decrypt XFiction. On the otherhand, in a *type II* attack, Eve would combine her attribute keys with a nonrevoked user (such as Bob) and try to decrypt XFiction. Not to mention CryptoFlix faces financial loss if any of the attacks becomes successful. This motivational example will be referred to repeatedly in the upcoming sections.

B. Limitations of the existing schemes and our novelty

Collusion resistance is a fundamental security requirement of any ABE scheme, as stated in the original ABE scheme proposed by Sahai et al. [2]. Initially proposed ABE schemes such as [1]–[3] do not support revocation. If CryptoFlix were to use such an ABE scheme, it could not revoke Eve even if her subscription expired. As a result, such schemes are not suitable for a practical application like secure cloud data sharing. They are resistant to *type I* collusion attacks. However, *type II* collusion attacks do not apply to them as those schemes do not support revocation. The revocation is a challenging problem in ABE since the same attribute may be shared among different users. Hur et al. proposed a solution to the revocation

This research is partially supported by a NSF grant 1449344.

problem in [4], [5]. The proposed solution is based on the idea of attribute group. The user's secret key consists of two parts. One is associated with the user's attributes, and the other is associated with the attribute group. They are called decryption secret key (DSK) and key encryption key (KEK), respectively. The revocation is dictated by KEK and is independent of the user's DSK. Consequently, the KEK of one user works with the DSK of another user. Hence, the proposed revocable ABE scheme is not resistant against *type II* collusion attacks as a revoked user by colluding with a nonrevoked user can get the valid KEK and restore his or her decryption ability. This vulnerability was first pointed out by Li et al. [6]. Schemes such as [7], [8] also have the same vulnerability since these solutions are also based on the same idea. Li et al. refined their initial solution [6] in [9]. To revoke a user from the attribute group, the attribute manager (AM) updates the existing user's KEK keys. They bind a user's DSK with his or her KEK so that the KEK of one user does not work with the DSK of another user. This ensures that a revoked user cannot collude with a nonrevoked user to restore his or her decryption right. However, this scheme has the following limitations:

- Each time a user is revoked, all nonrevoked users' keys (KEK) are affected because DSK and KEK keys are tied together by a common secret exponent that is only known to a semi-trusted party called the attribute manager (AM). This exponent is common across attribute secret keys of all the users. To revoke a user, this exponent needs to be updated in the secret keys for all nonrevoked users.
- It requires the additional semi-trusted AM for updating all nonrevoked users' secret keys (KEK) and distributing them to the respective users. This not only adds a lot of overhead, but also increases the security vulnerability by adding an additional semi-trusted party to the system.
- AM also becomes the performance bottleneck as it needs to participate in the key generation, key update, encryption, and re-encryption stages. Key generation is an one-time operation. However, other operations occur very frequently, which can be a huge burden for a centralized entity like AM to handle.

By following the footstep of [9], CryptCloud+ [10] also proposes a collusion-resistant revocable ABE scheme. In this scheme, the attribute authority has to periodically update the attribute secret keys of all the users according to a revocation list, which is very inefficient. Clearly, recent research leaves significant gaps. Our proposed revocable ABE scheme fills these research gaps for the first time. Like Li et al.'s schemes ([6] and [9]), our revocable ABE scheme is also collusion resistant against both *type I* and *type II* attacks. However, in our scheme, revocation does not affect the secret key of any nonrevoked user. Moreover, our scheme does not need the aid of any additional trusted entity, which minimizes the attack surface. This is made possible since we achieve revocation by modifying the core ABE secret key and ciphertext elements rather than achieving revocation by attribute group keys (KEK). We discuss our technique in the following section.

C. Our technique and contribution

Our revocation technique can be applied to any ABE scheme that does not support revocation (e.g., [1], [3], [11]). However, in this paper we choose the scheme proposed in [1] for several reasons. This scheme has a large universe construction (meaning that any arbitrary string can be used as an attribute) and it has been proven to be selectively secure in standard model (as opposed to the artificial random oracle model) under decisional q-Parallel Bilinear Diffie-Hellman exponent (decisional q-BDHE) assumption. These properties are more desirable in real-life application from both functional and security standpoints. Our ABE scheme also inherits these properties. We achieve revocation by modifying the core ABE secret key and ciphertext components and distributing them according to a binary tree. A data owner can revoke any user while encrypting a message using the minimum cover algorithm (Sec. III-F) on the binary tree. This required us to design new setup, encryption, keygen, and decryption algorithms for our proposed revocable ABE scheme. We briefly summarize our contribution as follows:

- We propose a collusion-resistant revocable ABE scheme that is resistant against both *type I* and *type II* collusion attacks. We achieve revocation property through modification of the core ABE secret key and ciphertext components. Hence, we eliminate the requirement of any semi-trusted entity for key updating.
- Revocation in our scheme never affects the secret keys of any nonrevoked user. A data owner can revoke any user's decryption ability from a particular file by including him/her in the revocation list during encryption.
- To show the effectiveness of our proposed collusion-resistant revocable ABE scheme, we build a secure cloud data sharing scheme based on it.
- Through proper security analysis, we prove that our scheme is collusion resistant.
- Through extensive theoretical and experimental performance analysis, we show that our scheme outperforms recently proposed similar schemes.

II. Related works

Revocable ABE was first addressed by Sahai et al. in [12], and then realized by Qin et al. in [13]. The attributes in both user's secret key and the access structure of a ciphertext are associated with different expiration time. The time specifies how long a secret key is allowed to decrypt a ciphertext. The cloud has to periodically update the access structure of a ciphertext using proxy re-encryption to enforce revocation. However, the problem is that, a colluding cloud can re-encrypt a ciphertext to an expiration time so that it can be decrypted by a user's expired secret key. The scheme proposed in [14] also incorporates the same idea of achieving revocation by updating the ciphertext. Instead of updating the ciphertext in a timely fashion, an aide server updates the ciphertext according to the data owner's provided revocation list. However, this scheme also suffers from a similar kind of collusion attack (server-revoked user) as [13]. Recently, [15] improved the security of [14] by replacing the single server with a dual server.

Hur et al. followed a different approach for revocation of ABE in [4] and later improved its security in [5]. According to their proposed solution, a user belongs to various attribute groups and the authorized set of attributes is determined by the attribute groups the user belongs to. A user's secret key consists of two parts: decryption secret key (DSK) and key encryption key (KEK), respectively. The DSK is associated

TABLE I: Comparison with related schemes in terms of security and functionality.

Scheme	Security assumption	Security Model	Collusion resistant	Revocation affects others' key
Hur-I [4]	Generic Group	RO	×	✓
Hur-II [5]	Generic Group	RO	×	✓
CryptCloud+ [10]	*l*-SDH	Standard	✓	✓
Flexible [6]	Generic Group	RO	✓	✓
UserCol [9]	Generic Group	RO	✓	✓
Ours	decisional q-BDHE	Standard	✓	×

with the user's attributes, while the KEK is associated with the user's attribute groups. Revocation is achieved by encrypting certain ciphertext components with KEK keys so that a user cannot decrypt the ciphertext without the appropriate KEK key despite having enough attributes in the DSK key. Li et al. first pointed out in [6] that DSK and KEK keys in Hur et al.'s proposed solution are independent of each other. As a result, a revoked user can collude with a nonrevoked user, and combine the nonrevoked user's KEK key with his or her own DSK keys to restore the decryption ability. Schemes like [7], [8] also suffer from the same revoked-nonrevoked user collusion attack. Li et al. proposed an initial solution to solve this collusion problem in [6], [16] and later improved the security in [9]. The revoked and nonrevoked user collusion problem was solved by binding the DSK key with the KEK key so that one user's DSK key does not work with another user's KEK key. The limitation of the proposed scheme was that a semi-trusted attribute manager has to update all nonrevoked users' KEK keys for revocation, which not only adds a lot of overhead, but also adds additional security vulnerability as the attribute manager can collude with revoked users to restroe their decryption ability. The attribute manager also needs to process every ciphertext, which can be a performance bottleneck. The solution proposed in CryptCloud+ [10] also relies on a semi-trusted entity for key update in revocation.

In Table I, we compare our scheme with the most relatable ones in terms of security and functionalities. Our scheme is selectively secure in the standard model under the decisional q-Parallel Bilinear Diffie-Hellman Exponent (decisional q-BDHE) assumption [1]. CryptCloud+ is proven to be secure in the standard model based on the hardness of *l*-Strong Diffie-Hellman (l-SDH) assumption. Hur-I [4], Hur-II [5], Flexible [6], and UserCol [9] are secure in the random oracle (RO) model with generic group assumption. According to [1] the random oracle model is an artificial model and not desirable for real-life applications. Hur-I and Hur-II are not resistant against a revoked-nonrevoked user collusion attack (*type II*), while the rest are secure against this attack. Note that only in our scheme does the revocation not affect the secret keys of the existing nonrevoked users.

III. Preliminaries

A. Symbols and Notations used

In this paper, we use $\mathbb{G}$ and $\mathbb{G}_T$ to represent two multiplicative cyclic groups of prime order p, while g is a generator of $\mathbb{G}$. The symbol $\mathbb{Z}_p$ is used to denote the group of integers modulo p. We also make use of a *randomness extractor function* [17] defined as $\mathcal{F} : \mathbb{G}_T \to \mathbb{K}$, where $\mathbb{K}$ is the symmetric key space. The encryption and decryption functions of the symmetric encryption scheme are denoted as *Enc* and *Dec*, respectively.

B. Bilinear Map

A bilinear map is a function e defined as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and must have the following properties:

1) *Bilinearity:* For $\forall g_1, g_2 \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_p$, the following relationship must always hold: $e\left(g_1^a, g_2^b\right) = e\left(g_1, g_2\right)^{ab}$
2) *Non-degeneracy:* $e\left(g_1, g_2\right) \neq 1$
3) *Computability:* group operations in $\mathbb{G}$ and e should be efficiently computable.

C. Decisional q-Parallel Bilinear Diffie-Hellman Exponent Assumption

We review the definition of decisional q-BDHE assumption from [1]. Following the notations from Sec. III-A, assume that a, s, and q exponents (e.g., $b_1, b_2, \ldots, b_q$) are randomly chosen from $\mathbb{Z}_p$. Then, according to the decisional q-BDHE assumption it is hard for any probabilistic polynomial time adversary to distinguish $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element $e(g,g)^r \in \mathbb{G}_T$ if the adversary is provided with the vector $\vec{v} = g, g^s, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, g^{2q}, \forall 1 \leq i \leq q : g^{sb_i}, g^{a/b_i}, \ldots, g^{a^q/b_i}, g^{a^{q+2}/b_i}, \ldots, g^{a^{2q}/b_i}, \forall 1 \leq i, j \leq q, i \neq j : g^{asb_j/b_i}, \ldots, g^{a^q sb_j/b_i}$. The advantage ϵ of a probabilistic polynomial time algorithm $\mathcal{B}$ in solving the decisional q-BDHE problem is defined as $\epsilon \leq |\Pr[\mathcal{B}(\vec{v}, A = e(g,g)^{a^{q+1}s}) = 0] - \Pr[\mathcal{B}(\vec{v}, A = e(g,g)^{a^r}) = 0]|$.

D. Access Structure

Let $\mathbb{P} = \{P_1, P_2, \cdot, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\mathbb{P}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\mathbb{P}$ (i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdot, P_n\}} \backslash \{\emptyset\}$). The sets in $\mathbb{A}$ are called authorized sets, and sets not in $\mathbb{A}$ are called unauthorized sets. In our context, the role of the parties is defined by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes.

E. Linear Secret Sharing Scheme (LSSS)

Let p be a large prime. Then, a secret sharing scheme $\prod$ over a set of parties P is called linear (over $\mathbb{Z}_p$) if

- The shares of each party form a vector over $\mathbb{Z}_p$.
- There exists a matrix M with l rows and n columns called the share-generating matrix for $\prod$. For all $i = 1, 2, \ldots, l$, the i^{th} row of M, we define a function ρ such that $\rho(i)$ maps row i of matrix M to an associated party. When we consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of l shares of the secret s according to $\prod$. The share $(Mv)_i$ belongs to party $\rho(i)$.

Any linear secret sharing scheme defined above has the following *linear reconstruction* property: Let $\prod$ be an LSSS for

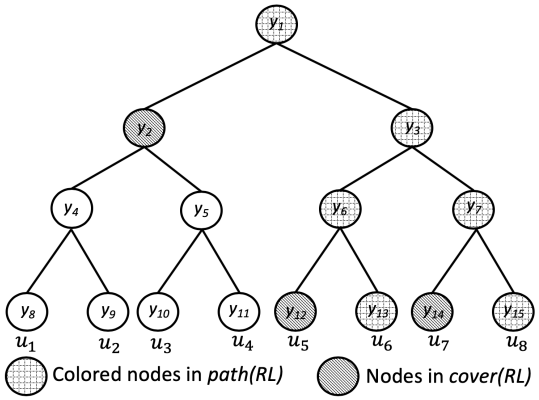

Fig. 1: Finding minimum cover

the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that if $\{\lambda_i\}$ are valid shares of any secret s according to $\prod$, then $\sum_{i \in I} w_i \lambda_i = s$. These constants $\{w_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

F. Minimum Cover

Let $\mathcal{T}$ be a full binary tree with m leaf nodes. Nodes in $\mathcal{T}$ are labelled as y_j, and each leaf node is associated with a user labelled as $u_1, u_2, \ldots, u_m$. We use *path(u_k)* to denote all the nodes in the path from the root to the associated leaf node of u_k. Let the set $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ and *RL* represent the set of all users and revoked users, respectively. Then, we can apply subset cover to get the minimum number of tree nodes (we call it *cover(RL)*) that cover $\mathcal{U} - RL$ (i.e., all nonrevoked users). The algorithm to find *cover(RL)* is as follows:

- $\forall y_j \in RL$, color all the nodes of *path(y_j)*.
- *cover(RL)* is the set of all uncolored nodes that are direct children of the colored nodes.

Figure 1 is an example of a binary tree $\mathcal{T}$ with 8 leaf nodes and 8 associated users (i.e., $\mathcal{U} = \{u_1, u_2, \ldots, u_8\}$). If *RL* = $\{y_{13}, y_{15}\}$, then *Cover(RL)* = $\{y_2, y_{12}, y_{14}\}$. Node that six nonrevoked users (i.e., $\{u_1, u_2, u_3, u_4, u_5, u_7\}$) are covered with just three nodes.

IV. System Architecture and Adversarial Model

A. System architecture

A high-level system architecture of our secure data sharing scheme has been presented in Fig. 2. Our architecture has four main entities: the attribute authority (AA), the cloud service provider (CSP), the data owner, and the data user. The attribute authority is responsible for managing all the attributes, and it creates and distributes attribute secret keys to the users according to their authorized attribute set. It also creates and publishes the public parameters. The CSP manages the public cloud where the encrypted data is outsourced and stored. Data owners encrypt their files using an attribute-based encryption scheme and uploads the files to the public cloud so that they are always available for the data users. Once the data is uploaded in the cloud, the data users can download and decrypt it anytime if they have enough attributes in their attribute secret keys. Note that a user may have the dual role of a data owner and a data user.

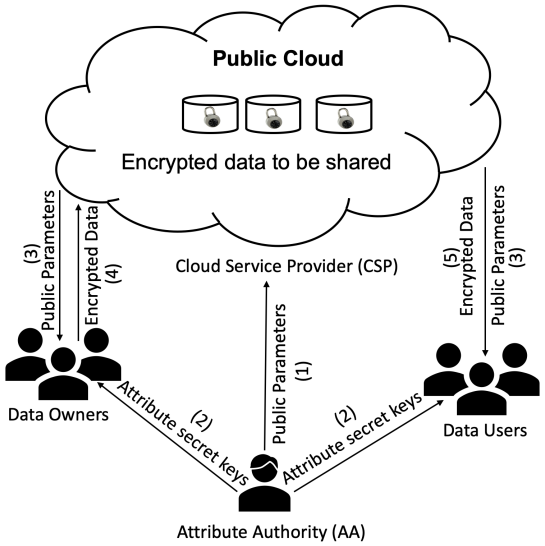

Fig. 2: System Architecture

B. Adversarial model

Security assumptions: We consider the CSP to be an honest but curious entity that is a standard practice in revocable ABE literature [5]–[7], [9], [18]. This implies that the CSP properly follows the protocols of our scheme (i.e., it honestly performs tasks like storing and updating encrypted files as per the data owners' request and letting the data users download encrypted files upon request). The CSP does not tamper with any stored information in the cloud. However, the CSP is open to deduce any plaintext information from the stored encrypted files and public parameters on its own. In previously proposed revocation schemes such as [4]–[6], [9], a semi-trusted entity (manager) needs to update users' secret keys and ciphertext in order to achieve revocation. This design introduces additional security vulnerabilities, as a compromised manager can update secret keys and ciphertext in a way that restores a revoked user's decryption right. Hence, user and manager collusion is not allowed. However, we do not need any semi-trusted manager to achieve revocation since the revocation right is given to the data owner, who can decide whom to revoke during encryption. We assume that the attribute authority is a trusted entity, and it distributes attribute secret keys to users via a secure channel such as SSL.

Adversaries and attacks: A dishonest data user is the main adversary of our system. A dishonest data user can be either revoked or nonrevoked. The goal of such adversaries is to decrypt a ciphertext that cannot be decrypted individually, either because they do not have enough attributes in their attribute secret keys or because they are revoked users (but may have enough attributes). To achieve this goal, a dishonest data user colludes with other dishonest data user(s) and launches *type I* and *type II* attacks.

V. Our Proposed Revocable ABE Scheme

There are two types of ABE schemes: ciphertext policy attribute-based encryption (CP-ABE) and key policy attribute-based encryption (KP-ABE). The access polity is embedded in the ciphertext in CP-ABE, while the access policy is embedded

in the keys in KP-ABE. We use ABE to denote CP-ABE in the rest of the paper unless otherwise specified. In this section, we first give the definition of our proposed revocable ABE scheme, followed by its security model. Finally, we give the detailed construction of our scheme.

A. Definition of our proposed revocable ABE

The revocable ABE scheme is consisted of four algorithms defined as follows:

• (PK, MK) ⟵ `Setup` ($Att_{\max}, l_{\max}, m$): The setup algorithm takes as input the maximum number of attributes allowed in a secret key, the maximum number of columns possible in a LSSS matrix, and the total number of users in the system denoted as $Att_{\max}$, $l_{\max}$, and m, respectively. It outputs the public key PK, and the master secret key MK.

• CT ⟵ `Encrypt` (PK, *(M, ρ)*, $\mathcal{M}$, *RL*): The inputs to the encryption algorithm are the public key PK, the LSSS access structure *(M, ρ)*, the message to be encrypted $\mathcal{M}$, and the revocation list *RL*. The algorithm outputs a ciphertext CT so that no user in *RL* can decrypt CT even if the attribute set S satisfies the access structure.

• SK ⟵ `Keygen` (PK, MK, *S*, u_k): The key generation algorithm takes as input the public key PK, the master secret key MK, the user's authorized attribute set S, and the user's identifier u_k. It outputs the user's secret key SK.

• $\mathcal{M}/\perp$ ⟵ `Decrypt` (PK, SK, CT): The decryption algorithm takes as input the public key PK, the user's secret key SK, and the ciphertext CT. It outputs the plaintext message $\mathcal{M}$ if the user does not belong to the corresponding revocation list *RL* of CT and his or her authorized attribute set S satisfies the access structure. The decryption algorithm outputs $\perp$ otherwise.

B. Security model

We formalize the security model of our proposed revocable ABE scheme by the following IND-CPA (indistinguishable chosen plaintext attack) game.

Init. The adversary $\mathcal{A}$ commits to an access structure (M^*, ρ) by giving it to the challenger.

Setup. The challenger runs the `Setup` ($Att_{\max}, l_{\max}, m$) algorithm to generate PK, SK, and sends PK to $\mathcal{A}$.

Phase I Query. The adversary $\mathcal{A}$ repeatedly makes $q1$ private key queries for the user-authorized attribute set tuples as in $Q_1 = (u_1, S_1), Q_2 = (u_2, S_2), \ldots, Q_{q1} = (u_{q1}, S_{q1})$. The challenger calls `Keygen` (PK, MK, S_k, u_k) for each query $Q_k = (u_k, S_k)$, and sends the secret key SK_k to $\mathcal{A}$.

Challenge. $\mathcal{A}$ selects two equal size messages $\mathcal{M}_0, \mathcal{M}_1$, a revocation list RL^*, and sends them to the challenger. Additionally, $\mathcal{A}$ also sends to the challenger the committed access structure (M^*, ρ). The challenger chooses a bit $b \in \{0, 1\}$ by flipping a random coin, and runs `Encrypt` (PK, (M^*, ρ), $\mathcal{M}_b$, RL^*). The challenger then sends the output ciphertext CT^* to $\mathcal{A}$. The constraint is that none of the attribute sets (e.g., $S_1, S_2, \ldots, S_{q1}$) in phase I query satisfy the access structure (M^*, ρ).

Phase II Query. $\mathcal{A}$ adaptively makes private key queries for tuples $Q_{q1+1} = (u_{q1+1}, S_{q1+1}), Q_{q1+2} = (u_{q1+2}, S_{q1+2}), \ldots, Q_q = (u_q, S_q)$ with the restriction that none of the attribute sets in these tuples (e.g., $S_{q1+1}, S_{q1+2}, \ldots, S_q$) satisfy the committed access structure (M^*, ρ).

Guess. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ for b.

The advantage of the adversary $\mathcal{A}$ in the above game is defined as $Adv = |\Pr[b' = b] - 1/2|$. By allowing $\mathcal{A}$ to do decryption queries in phase I and phase II query stage, this security model can be easily extended to chosen-ciphertext attack (CCA).

Definition 1: Our proposed revocable ABE scheme is secure if all polynomial time adversaries have at most a negligible advantage in the IND-CPA game.

C. Construction of Our Proposed Revocable ABE Scheme

The detailed construction of our proposed collusion-resistant revocable ABE scheme is given as follows:

• (PK, MK) ⟵ `Setup` ($Att_{\max}, l_{\max}, m$): The setup algorithm takes as input $Att_{\max}$, the maximum number of attributes any user's secret key may have; $l_{\max}$, the maximum number of columns any LSSS matrix M may have; and m, the total number of users. It outputs the public and the master secret key PK and SK, respectively. The setup algorithm first chooses a group $\mathbb{G}$ of prime order p with a generator g and defines a bilinear map as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We assume that attributes can be represented in $\mathbb{Z}_p$. In practice, a collision-resistant hash function can be used to transform any string attribute into $\mathbb{Z}_p$. The setup algorithm then randomly chooses $a, \alpha, \beta \in \mathbb{Z}_p$. It also utilizes a hash function defined as $\mathcal{H} : \mathbb{Z}_p \to \mathbb{G}$. The hash function is realized by choosing a polynomial $\mathcal{L}(x) \in \mathbb{Z}_p$ of degree $N = Att_{\max} + l_{\max} - 1$ and computing $h_0 = \mathcal{H}(0) = g^{\mathcal{L}(0)}, h_1 = \mathcal{H}(1) = g^{\mathcal{L}(1)}, \ldots, h_N = \mathcal{H}(N) = g^{\mathcal{L}(N)}$. With these $N + 1$ values, one can compute $h_x = \mathcal{H}(x) = g^{\mathcal{L}(x)}$ for any $x \in \mathbb{Z}_p$ by using interpolation.

The AA then creates a full binary tree $\mathcal{T}$ of m leaves and associates each user u_j to a different leaf node. For each node y_i in $\mathcal{T}$, AA randomly chooses $g_{y_i} \in \mathbb{G}$. Finally, AA publishes the public key as $PK = \left(\mathcal{T}, \mathbb{G}, \mathbb{G}_T, e(g, g)^\alpha, g^a, g^\beta, h_0, h_1, \ldots, h_N\right)$ and sets the master secret key as $MK = (g^\alpha, \beta)$ and keeps it secret.

• CT ⟵ `Encrypt` (PK, *(M, ρ)*, $\mathcal{M}$, *RL*): The encrypt algorithm takes as input the public parameters PK, LSSS access structure *(M, ρ)*, the plaintext message $\mathcal{M} \in \mathbb{G}_T$, and a revocation list *RL*. It outputs the ciphertext (CT). In the LSSS access structure *(M, ρ)*, *M* is an $l \times n$ matrix and ρ is a function that associates rows of *M* to attributes. In this construction, ρ is limited to be an injective function (i.e., an attribute can be associated with at most one row of *M*). The algorithm chooses a random vector $\vec{v} = (s, r_2, \ldots, r_n) \in \mathbb{Z}_p^n$. These values are used to share the random encryption exponent S. For $\forall i \in \{1, 2, \ldots, l\}$ it computes $\lambda_i = \vec{v}.M_i$, where M_i is the vector corresponding to the i^{th} row of M. Finally, the algorithm finds *cover(RL)*, randomly chooses an exponent $r \in \mathbb{Z}_p$, and computes the ciphertext as $CT = (C = \mathcal{M}.e(g, g)^{\alpha s}, C' = g^{s\beta}, D = g^r, \forall y_j \in \textit{cover(RL)} : C_{y_j} = g_{y_j}^s, \forall i \in \{1, 2, \ldots, l\} : C_i = g^{a\lambda_i}\mathcal{H}(\rho(i))^{-r})$. We assume that the LSSS access structure is implicitely included in CT.

• SK ⟵ `Keygen` (PK, MK, *S*, u_k): The keygen algorithm takes as input PK, MK, a set *S* of attributes the user is authorized for, and the user's identifier u_k and outputs the attribute secret key SK. Let u_k's associated leaf node in $\mathcal{T}$ be y. The algorithm finds *path(y)*, chooses a random $t \in \mathbb{Z}_p$, and creates the private key as $SK = (\forall y_j \in \textit{path(y)} : K_{y_j} = \left(g^{\alpha + at} g_{y_j}\right)^{1/\beta}, L = g^t, \forall x \in S : K_x = \mathcal{H}(x)^t)$.

• $\mathcal{M}/\perp \longleftarrow$ `Decrypt` (PK, SK, CT): The decryption algorithm takes as input the public paprameters PK, the user's attribute secret key SK for an attribute set *S*, and the ciphertext CT for a LSSS access structure *(M, ρ)*. Assume that S satisfies the access structure and $I \subset \{1, 2, 3, \ldots, l\}$ is defined as $I = \{i : \rho(i) \in S\}$. The algorithm finds the constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that if $\{\lambda_i\}$ are valid shares of a secret *s* according to *M*, then $\sum_{i \in I} w_i \lambda_i = s$. Note that these constants $\{w_i\}$ can be found in polynomial time in the size of *M* and there could potentially be different ways of choosing such $\{w_i\}$. The algorithm then computes the following:

$$
\begin{aligned}
P &= e(K_{y_j}, C') = e\left(\left(g^{\alpha + at} g_{y_j}\right)^{1/\beta}, g^{\beta s}\right) \\
&= e(g,g)^{\alpha s + ats}.e(g_{y_j}, g)^s \\
Q &= e(C_{y_j}, g) = e(g_{y_j}, g)^s \\
W &= P/\left(Q \prod_{i \in I} \left(e(C_i, L) e(D, K_{\rho(i)})\right)^{w_i}\right) \\
&= P/\left(Q \prod_{i \in I} e(g,g)^{at \lambda_i w_i}\right) \\
&= e(g,g)^{\alpha s + ats}.e(g_{y_j}, g)^s / \left(e(g_{y_j}, g)^s e(g,g)^{ats}\right) \\
&= e(g,g)^{\alpha s}.
\end{aligned}
$$

Then, the decryption algorithm retrieves the plaintext message $\mathcal{M}$ as in $C/W = \mathcal{M}$, and outputs it. If S does not satisfy the access structure, or the owner of SK belongs to the revocation list *RL*, then the decryption algorithm outputs $\perp$.

VI. A Secure cloud data-sharing scheme based on our proposed ABE scheme

In this section, we propose a secure data-sharing scheme for the cloud based on our proposed collusion-resistant revocable ABE scheme. We discuss the details in the following.

A. System initialization

The attribute authority (AA) runs the `Setup` algorithm to initialize the system. It publishes the public key PK in the cloud and keeps the master key MK secret.

B. Key distribution

For each user in the system, the AA runs the `Keygen` algorithm and generates the attribute secret key SK for his or her authorized attribute set *S*. The AA then sends SK to the corresponding user via a secure channel.

C. File outsourcing

The data owner downloads the public parameter PK from the cloud. It randomly chooses $\mathcal{M} \in \mathbb{G}_T$ and extracts the symmetric key using the *randomness extractor function* $\mathcal{F}$ as in $\mathcal{K} = \mathcal{F}(\mathcal{M})$. Using $\mathcal{K}$, the data owner encrypts the actual file F using the symmetric encryption scheme as in CT′ =*Enc*($\mathcal{K}$, *F)*. Then, the data owner includes any of the users in the revocation list *RL* to revoke from the ciphertext, runs `Encrypt` (PK, *(M, ρ)*, $\mathcal{M}$, *RL*) algorithm, and receives CT as output. The data owner then uploads the final ciphertext $\left(\text{CT}, \text{CT}'\right)$ as an encrypted file in the cloud.

D. File retrieving

The data user downloads an encrypted file $\left(\text{CT}, \text{CT}'\right)$ from the cloud and calls `Decrypt` (PK, SK, CT) algorithm with his or her attribute secret key SK. If SK has enough attributes to satisfy the access policy of CT, the algorithm returns $\mathcal{M}$ as an output. Then, the user extracts the symmetric key using the *randomness extractor function* $\mathcal{F}$ as in $\mathcal{K} = \mathcal{F}(\mathcal{M})$. Using $\mathcal{K}$, the data user retrieves the actual file F by running the decryption function of the symmetric encryption scheme as in F =*Dec*($\mathcal{K}$, *CT′)*.

VII. Security Analysis

In this section, we analyze the security of our proposed revocable ABE scheme in terms of semantic security and collusion attacks.

A. Semantic security

Our IND-CPA game (in Sec. V-B) is similar to that of [1] except in our game, each secret key query in query phase I and II also includes a user identifier u_i, and $\mathcal{A}$ provides with a revocation list *RL** in the challenge stage. It was proven in [1] that if the decisional *q*-BDHE assumption holds, then no polynomial-time adversary $\mathcal{A}$ can selectively win the IND-CPA game with a challenge matrix M^* of size $l^* \times n^*$ corresponding to the access structure (*M**, ρ), and maximum number of attributes per key of *Att*$_{\max}$ where $n^* + Att_{\max} \leq q$.

Assume there exists an adversary $\mathcal{A}$ who has a non-negligible advantage $\epsilon = \text{Adv}_{\mathcal{A}}$ in the our IND-CPA game and it chooses a challenge access structure (*M**, ρ) with the matrix M^* of at most q columns. Then using the same technique as in [1], we can build a simulator $\mathcal{B}$ that plays the decisional *q*-BDHE problem. The simulator basically programs all the IND-CPA game parameters from the decisional *q*-BDHE parameters so that the challenger cannot distinguish whether it is playing our IND-CPA game or the decisional *q*-BDHE problem. Since $\mathcal{A}$ has a non-negligible advantage in our IND-CPA game (according to our prior assumption), $\mathcal{A}$ also has a non-negligible advantage in the decisional *q*-BDHE problem. However, according to the definition (of decisional *q*-BDHE problem), no polynomial-time adversary has a non-negligible advantage in solving the decisional *q*-BDHE problem. This implies that $\mathcal{A}$ does not have a non-negligible advantage in our IND-CPA game. So, the DEFINITION 1 holds true.

B. Collusion attack

The revocable ABE needs to be secure against both *type I* and *type II* collusion attack. The semantic security discussed earlier also guarantees that our scheme is secure against *type I* collusion attack since the secret key queries made by the adversary $\mathcal{A}$ in query phase I and phase II cannot individually satisfy the committed access structure (*M**, ρ). In this section, we formalize the security of our scheme against *type II* collusion attack by the following theorem.

Theorem 1: A revoked user with enough attributes in his or her attribute secret key cannot decrypt a ciphertext even if the user colludes with a nonrevoked user who does not have enough attributes in his or her attribute secret key to decrypt the particular ciphertext.

Proof: Assume that u_i, u_k are two users and y, y' are their associated leaf nodes in the binary tree. SK = $(\forall y_j \in$

TABLE II: Symbols used in performance analysis and experiment

Symbol	Meaning
u	Total number of attributes in the system
a	Number of attributes in access structure $\mathbb{A}$
b	Number of attributes in user secret key
s	Required minimum number of attributes to satisfy policy
$\|\mathbb{G}_i\|$	Size of a single element in group $\mathbb{G}_i$
$\|\mathbb{K}\|$	Size of symmetric key
$\|p\|$	Size of a single element in $\mathbb{Z}_p$
C_i	Single exponentiation time in group $\mathbb{G}_i$
P	Computation time of a pairing operation
m	Total number of users in the group
r	Total nodes in *cover(RL)*

path(y) : $K_{y_j} = \left(g^{\alpha+at} g_{y_j}\right)^{1/\beta}, L = g^t, \forall x \in S : K_x = \mathcal{H}(x)^t)$ and $\text{SK}' = (\forall y_j \in \textit{path}(y') : K'_{y_j} = \left(g^{\alpha+at'} g_{y_j}\right)^{1/\beta}, L' = g^{t'}, \forall x \in S' : K'_x = \mathcal{H}(x)^{t'})$ are their attribute secret keys, respectively. A message $\mathcal{M}$ is encrypted so that $u_i \in$ *RL*. This results in a ciphertext $\text{CT} = (C = \mathcal{M}.e(g,g)^{\alpha s}, C' = g^{s\beta}, D = g^r, \forall y_j \in \textit{cover(RL)} : C_{y_j} = g^s_{y_j}, \forall i \in \{1,2,\ldots,l\} : C_i = g^{a\lambda_i}\mathcal{H}(\rho(i))^{-r})$. Note that CT does not have any C_{y_j} corresponding to the attribute secret key component K_{y_j} of u_i since $\textit{path}(y) \cap \textit{cover(RL)} = \emptyset$. However, there exists a component K'_{y_j} in the attribute secret key of the nonrevoked user u_k that corresponds to a C_{y_j} since $\textit{path}(y') \cap \textit{cover(RL)} \neq \emptyset$. Without the lose of generality, lets assume that S satisfies the access structure (M, ρ) but S' does not satisfy (M, ρ). This implies that u_i has enough attributes in his or her attribute secret keys to decrypt CT but u_k does not have enough attributes to do so. In order to successfully decrypt CT, u_i also needs K_{y_j} in his or her attribute secret keys that corresponds to C_{y_j}. As a result, u_i alone cannot decrypt CT with SK despite having enough attributes in it. However, u_i can collude with the nonrevoked user u_k to get K'_{y_j} from u_k's attribute secret key SK' that corresponds to a C_{y_j} in CT. Then, u_i can try to decrypt CT as follows:

$$
\begin{aligned}
P &= e(K'_{y_j}, C') = e\left(\left(g^{\alpha+at'} g_{y_j}\right)^{1/\beta}, g^{\beta s}\right) \\
&= e(g,g)^{\alpha s + at' s}.e(g_{y_j}, g)^s \\
Q &= e(C_{y_j}, g) = e(g_{y_j}, g)^s \\
W' &= P / \left(Q \prod_{i \in I} \left(e(C_i, L) e(D, K_{\rho(i)})\right)^{w_i}\right) \\
&= P / \left(Q \prod_{i \in I} e(g,g)^{at\lambda_i w_i}\right) \\
&= e(g,g)^{\alpha s + at' s}.e(g_{y_j}, g)^s / \left(e(g_{y_j}, g)^s e(g,g)^{ats}\right) \\
&= e(g,g)^{\alpha s + at' s - ats}.
\end{aligned}
$$

It is apparent that u_i cannot successfully compute $W = e(g,g)^{\alpha s}$ and hence is unable to retrieve $\mathcal{M}$ from $C = \mathcal{M} e(g,g)^{\alpha s}$. This proves THEOREM 1.

VIII. Theoretical Performance Analysis

In this section, we compare our proposed ABE scheme with other related schemes in terms of storage, communication, and computational efficiency from the theoretical aspect. Note that we also include [1] (referred to as BW) in the comparison as a baseline since our scheme is based on this. Table II illustrates different symbols that have been used for this purpose.

A. Storage and Communication Efficiency

The space efficiency comparison in terms of ciphertext, secret key, and public key size has been summarized in Table III. The ciphertext size, secret key size, and public key size represent the storage cost required by the cloud, each user, and the attribute authority to store them, respectively. Additionally, they represent the communication cost when these are sent from one party to another. However, we do not consider here the communication cost that is associated with any intermediate step during the preparation of the ciphertext, secret key, or public key. For example, in order to achieve revocation, the data owner sends the whole ciphertext to the manager for re-encryption in UserCol [9]. As a result, the communication cost for sending the ciphertext to the cloud would be twice as much as what is shown in Table III for [9]. Similar intermediate steps are necessary in [5], [6], [9] during the secret key generation phase. Thus, the actual communication cost can be higher than what is shown in Table III. However, there is no intermediate step in our proposed scheme, so the cost shown in Table III for our scheme is much closer to the real cost.

Compared to the baseline scheme [1], our scheme requires $r+1$ and $\log m - 1$ additional group ($\mathbb{G}_1$) elements for the ciphertext and secret key, respectively. This is because the data owner has to create $r+1$ additional group elements in the ciphertext out of which r elements are for *cover(RL)*. On the other hand, the AA has to create $\log m - 1$ additional group elements in the user's secret key along the *path* in the binary tree. Our scheme also needs $2m-1$ additional group elements in the public parameter compared to the baseline scheme because a group element ($\mathbb{G}_1$) associated with every node in the binary tree is needed in the public parameter.

Among all collusion-resistant revocable schemes (ours and [6], [9], [10]), [9] has the biggest ciphertext because for each attribute present in the policy of the ciphertext, r additional group elements are needed. This results in a total of ra additional group elements in the ciphertext. How our scheme compares against [6], [10] in terms of ciphertext size, depends on the value of total number of attributes in the ciphertext (a) and the number of nodes in *cover(RL)* (r). If there are few members to revoke or if revoked members are not very sparsely distributed in the binary tree, then r will be much smaller, and hence our scheme will have smaller ciphertext. For bigger policies (i.e., bigger a), our ciphertext size will be relatively smaller. In terms of the secret key size, our scheme requires at most $\log m$ additional group elements compared to [6], [9], [10]. However, if the attribute secret key holds a lot of attributes, then [9] will have larger secret key size. For example, if there are 1000 users in the group and a user has 10 attributes in his or her attribute secret key, then there will be only 21 group elements in our secret key as opposed to 41 in [9]. Public key size increases proportionally with the number of total users for both our scheme and [9]. However, the total public parameter size of [9] is larger than ours.

Compared to [4] and [5] (not resistant against *type II* collusion attacks), our scheme has a larger public key size, as the public key includes an additional group element for each node in the binary tree. While our scheme requires $\log m$ additional

TABLE III: Comparison of storage and communication efficiency with other schemes

Scheme	Ciphertext size	Secret key size	Public key size
Hur-I [4]	$(2a+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(2b+1)\|\mathbb{G}_1\| + (\log m)\|\mathbb{K}\|$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
Hur-II [5]	$(2a+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$2(b+1)\|\mathbb{G}_1$	$3\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
CryptCloud+ [10]	$(2a+5)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(b+6)\|\mathbb{G}_1\| + 2\|p\|$	$(u+6)\|\mathbb{G}_1\| + 3\|p\|$
Flexible [6]	$(2a+6)\|\mathbb{G}_1\| + \|\mathbb{G}_T\| + 2\|p\|$	$(b+4)\|\mathbb{G}_1\| + 2\|p\|$	$3\|\mathbb{G}_1\| + 2\|\mathbb{G}_T\| + \|p\|$
UserCol [9]	$(2a+ra+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$4b\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$2(u+3)\|\mathbb{G}_1\| + 2\|\mathbb{G}_T\| + (2m-1)\|p\|$
BW [1]	$(a+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(b+2)\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
Ours	$(a+r+2)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(b+1+\log m)\|\mathbb{G}_1\|$	$(2m+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$

group elements, [4] requires $\log m$ additional symmetric keys in the secret key because it includes $\log m$ KEK keys to each user's secret key along the *path* of the associated leaf node in the KEK key tree. However, [5] improves this by adding only a single group element per attribute in the secret key. How [4] and [5] compare against our scheme in terms of ciphertext size is similar to the logic we presented earlier while comparing our scheme with [6], [10].

B. Computation cost analysis

We show the computation cost of our scheme and compare it with other schemes in Table IV. The computation cost has been expressed in terms of group exponentiation and pairing operation in a similar manner as in [4], [5], [17]. This is a reasonable consideration since these two operations dominate relatively lightweight hash, multiplication, division, and addition operations.

Compared to the baseline scheme [1], our scheme requires just one additional pairing operation (in $\mathbb{G}_1$) for decryption. For encryption, our scheme requires $r+1$ additional group exponentiation operations (in $\mathbb{G}_1$), out of which r is for creating r additional group elements for *cover(RL)*. On the other hand, our scheme requires $\log m$ additional group exponentiation operations (in $\mathbb{G}_1$) for the secret key generation. Among $\log m$ additional group operations (in $\mathbb{G}_1$), $\log m - 1$ is for creating $\log m - 1$ additional group elements in the user's secret key along the *path* of the associated leaf node in the binary tree.

Our key update cost is zero since we achieve revocation without affecting the secret key of other nonrevoked users. In contrast, all other revocation schemes require a significant amount of computation for key updating to achieve revocation. The number of group exponentiation operation (in $\mathbb{G}_1$) required for key update is proportional to the number of users in the system (except [4]).

In terms of decryption speed, our scheme outperforms all the revocation scheme, thanks to only one additional pairing operation for decryption compared to the baseline scheme [3]. We can see that [5] has the slowest decryption speed as the required number of group exponentiation operation (in $\mathbb{G}_1$) is proportional to the total number of users (m).

The number of group exponentiation operation (in $\mathbb{G}_1$) for our key generation algorithm increases logarithmically with the number of users (m). However, our key generation algorithm is still faster than that of [10]. This is because in addition to the group exponentiation operation, CryptCloud+ also requires $(2b+7)P$ pairing operations, which is more expensive than the group exponentiation operation. Our key generation cost may even be lower than that of [4]–[6], [9] if the number of attributes in the secret key (b) is relatively high since $\log m$ increases very slowly.

Our encryption speed is much faster than that of [5], [9] since the required number of group exponentiation operation (in $\mathbb{G}_1$) is proportional to r and ra for [5] and [9], respectively. Among all the revocation schemes, [10] has the fastest encryption time.

IX. Experiment

Implementation: We have implemented our scheme in `Charm` [19]. It is a Python-based framework developed for rapid prototyping of advanced cryptographic protocols. `Charm` uses PBC library [20] (written in C language) for low-level system calls including most expensive group exponentiation and pairing operations. As a result, cryptographic protocols written in `Charm` performs very close to the one written C language [21]. All hash functions were implemented using SHA224.

A detailed parameter description for our experimental setup is given in Table V. SS512 is a super singular EC curve (with symmetric Type 1 pairing), and MNT224 is the Miyaji, Nakabayashi, Takano curves (with asymmetric Type 3 pairing). In Table II, p = bit length of prime order p, k = embedding degree, *Security* is the security level in bits with respect to the discrete log problem, and the numbers associated with the curve name represent the base field size in bits (i.e., SS512 has a base field size of 512 bits). Though our ReVO-ABE construction is based on a symmetric pairing group ($\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$), we have tested our implementation in both symmetric and asymmetric group settings. `Charm` treats groups as asymmetric, though the actual setting depends on the type of underlying chosen curve. More specifically, there are three different groups ($\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$), and pairing is defined as $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We keep most of the terms in $\mathbb{G}_1$ while implementing our scheme in the asymmetric setting since operations in $\mathbb{G}_1$ are generally much faster than those in $\mathbb{G}_2$.

Testbed setup: We have conducted all the experiments on a Macbook Pro laptop with Intel® Core i7@2.2 GHz quad-core processor and 16 GB RAM running MacOS 10.14.6. We have used Python 3.7 and the PBC-0.5.14 library.

Results: We have mainly compared our key generation, encryption, decryptioin, and key update running time with CryptoCloud+ [10], Flexible [6], and UserCol [9]. We have also reported the key generation, encryption, and decryption running time of BW [1] scheme as a baseline comparison since our scheme is based on the BW scheme. However, we have not compared our experimental results with Hur I [4] and Hur II [5] since they are not collusion resistant (against *type II* attacks) and the encryption, decryption, and key update of Hur II takes much longer than the rest. The running time of each algorithm (key generation, encryption, decryption, and key update) were measured in SS512 and MNT224 curves. Each

TABLE IV: Comparison with other schemes in terms of computation cost.

Scheme	**Key generation**	**Encryption**	**Decryption**	**Key update**
Hur-I [4]	$2(b+1)C_1$	$(3a+1)C_1 + C_T$	$C_1 + (2s-1)C_T + (2s+1)P$	bC_1
Hur-II [5]	$(3b+5)C_1$	$(3a+2m+3)C_1 + C_T$	$s(m+1)C_1 + (2s-1)C_T + (3s+1)P$	bmC_1
CryptCloud+ [10]	$(b+13)C_1 + C_T + (2b+7)P$	$(a+5)C_1 + C_T$	$2C_1 + sC_T + (2s+5)P$	$3mC_1$
Flexible [6]	$(2b+9)C_1 + 2P$	$2(a+3)C_1 + 2C_T$	$(2s+3)C_T + (2s+4)P$	$(2m+b+1)C_1 + P$
UserCol [9]	$(4b+2)C_1$	$(3a+ra+1)C_1 + C_T$	$(2s-1)C_T + (3s+1)P$	$(2m-1)C_1$
BW [1]	$(b+2)C_1$	$(2a+1)C_1 + C_T$	$sC_T + (2s+1)P$	N/A
Ours	$(b+2+\log m)C_1$	$(2a+r+2)C_1 + C_T$	$sC_T + 2(s+1)P$	0

TABLE V: Parameter details for different pairing groups

Curves	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	p	k	*Security*
SS512	512	512	1024	160	2	80
MNT224	224	672	1344	224	6	100

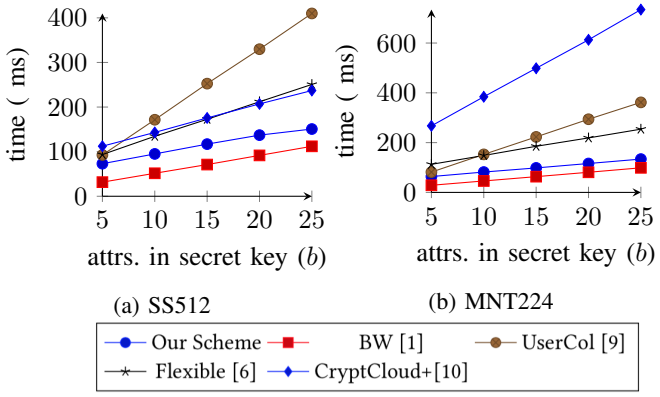

Fig. 3: Key generation time

result reported here has been averaged over five individual runs.

The running time of the key generation algorithm linearly increases with the total number of attributes in the secret key (b) (Fig. 3). However, our scheme also has a logarithmic relationship with the total number of users (m). We have set the value of m to be 1000 and measured the running time of key generation algorithm by varying the value of b between 5 and 25. For all four curves, our running time is close to that of BW. Even for the value of b being as little as 5, our running time is lower than that of others. The performance of our key generation algorithm is even better compared to others for a higher value of b because with the higher value of b, the number of group exponentiation operation (in $\mathbb{G}_1$) increases at a slower rate compared to others (refer to Table IV). Interestingly, the running time of UserCol is the highest for the SS512 curve (Fig. 3a). However, for MNT224 curve, its running time is much faster than that of CryptCloud+ because the pairing operations take more time in MNT224 curve compared to the SS512 curve, and CryptCloud+ requires $(2b+7)$ pairing operations while UserCol requires only two pairing operations for the key generation.

The encryption running time has been shown in Fig. 4. UserCol has the fastest running time and CryptCloud+ has the slowest running time. The running time of encryption algorithm mainly depends on the access structure length or the number of attributes in the access structure (a). This is because the number of group exponentiation operations (in $\mathbb{G}_1$) has a linear relationship with a. However, the number of

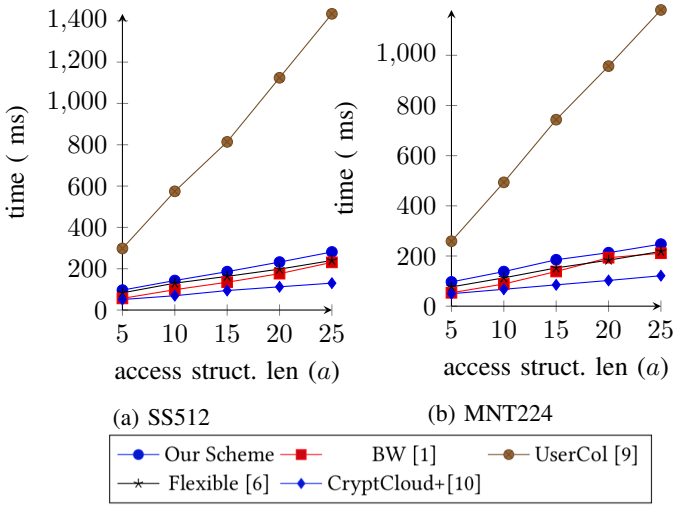

Fig. 4: Encryption time

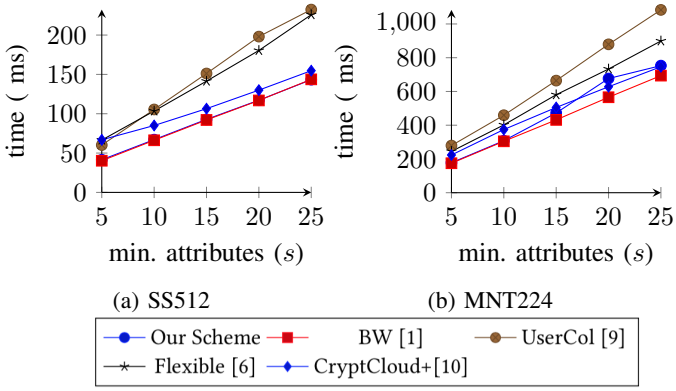

Fig. 5: Decryption time

group exponentiation operations of our scheme and UserCol is also proportional to r and ar, respectively. As a result, the running time of UserCol is significantly faster than the rest. For our experiment, we have set the value of r to be 10 and varied the value of a between 5 and 25.

Fig. 5 shows that the running time of the decryption algorithm has a linear relationship with the number of minimum required attributes to satisfy the access structure (s). We have varied the value of s from 5 to 25 at an interval of 5. Both the number of group exponentiation (in $\mathbb{G}_T$) and pairing operation has a linear relationship with s for all schemes. The pairing operation is much slower in MNT224 curve compared to the SS512 curve. Consequently, the decryption running time is higher in the MNT224 curve for all schemes. Note that

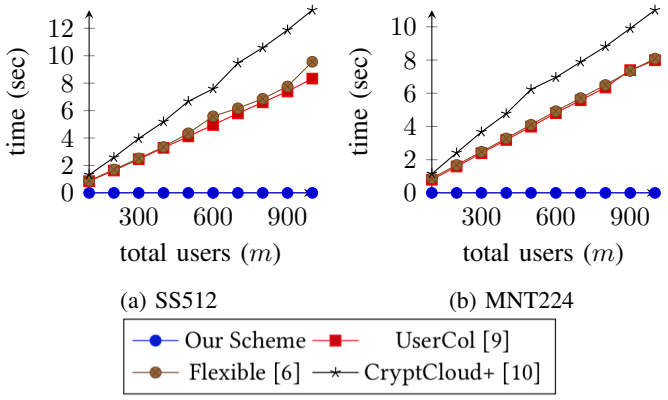

Fig. 6: Key update time

our decryption time is very close to the baseline scheme, as our decryption requires only one additional pairing operation compared to that of the baseline scheme.

The revocation in our scheme does not affect the secret key of any nonrevoked user. As a result, our revocation does not require any key update and hence the key update cost is zero for our scheme. However, other revocable schemes need to update the secret keys of existing nonrevoked users. In Fig. 6, we have shown how the running time of the key update algorithm increases with the number of users (m) by varying the value of m from 100 to 1000. The running time of the key update algorithm increases linearly with m for all schemes except ours. In this comparison we exclude [1] as it does not support revocation, and hence the key update time is irrelevant.

Note that, the performance of all the algorithms do not have equal importance. For instance, key generation is normally a one time task. A file may be encrypted by the owner only once but is potentially decrypted many times by different users. If users are revoked frequently, then key update cost can be very critical. As a result, the performance of decryption and key update are more important than that of other algorithms (e.g., key generation and encryption). From Fig. 5 and 6 we can see that our decryption and key update algorithms outperform other schemes.

X. Conclusion and Future Works

In this paper, we have proposed a revocable ABE scheme that is resistant against both *type I* and *type II* collusion attacks. Our scheme does not require any semi-trusted entity to achieve revocation. Moreover, the revocation does not affect the secret key of any non-revoked user, and hence the key update cost for revocation is zero in our scheme. We have also proposed a cloud-based secure data sharing scheme based on our proposed revocable ABE. Through security analysis, we have shown that our scheme is collusion (both *type I* and *type II*) resistant. It is evident from both theoretical performance analysis and experimental results that our scheme outperforms the most relatable ABE schemes. However, in this work we have only considered user-level revocation, but not attribute-level revocation. More specifically, using our proposed revocation scheme, a user (e.g., Eve) can be revoked from a ciphertext as a whole, but not some of his or her specific attributes (e.g., revoke only *new_release* and *movie* while keeping *tv_show, documentary,* and *scifi* intact). In the future, we want to extend our work to support attribute-level revocation. We have also limited the scope of this work to static access structures only. It will be an interesting extension if we could incorporate dynamic access policies in our scheme in the future.

References

[1] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*, pp. 53–70, Springer, 2011.

[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, 2005.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*, pp. 321–334, IEEE, 2007.

[4] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[5] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 10, pp. 2271–2282, 2013.

[6] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, no. 5, pp. 785–796, 2017.

[7] H. Wang, Z. Zheng, L. Wu, and P. Li, "New directly revocable attribute-based encryption scheme and its application in cloud storage environment," *Cluster Computing*, vol. 20, no. 3, pp. 2385–2392, 2017.

[8] X. Wang and J. Fang, "A revocable outsourcing attribute-based encryption scheme," in *Cloud Computing, Security, Privacy in New Computing Environments: 7th International Conference, CloudComp 2016, and First International Conference, SPNCE 2016, Guangzhou, China, November 25–26, and December 15–16, 2016, Proceedings*, vol. 197, p. 145, Springer, 2017.

[9] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance cp-abe with efficient attribute revocation for cloud storage," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1767–1777, 2018.

[10] 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 Transactions on Services Computing*, 2018.

[11] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, "Ciphertext-policy attribute-based encryption with key-delegation abuse resistance," in *Australasian Conference on Information Security and Privacy*, pp. 477–494, Springer, 2016.

[12] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Advances in Cryptology–CRYPTO 2012*, pp. 199–217, Springer, 2012.

[13] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, pp. 355–370, 2014.

[14] G. Zeng, "Server-aided directly revocable ciphertext-policy attribute-based encryption with verifiable delegation," in *Information and Communications Security: 19th International Conference, ICICS 2017, Beijing, China, December 6-8, 2017, Proceedings*, vol. 10631, p. 172, Springer, 2018.

[15] B. Qin, Q. Zhao, D. Zheng, and H. Cui, "(dual) server-aided revocable attribute-based encryption with decryption key exposure resistance," *Information Sciences*, vol. 490, pp. 74–92, 2019.

[16] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *International Journal of Communication Systems*, vol. 30, no. 1, p. e2942, 2017.

[17] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1384–1393, 2015.

[18] R. Zhang, L. Hui, S. Yiu, X. Yu, Z. Liu, and Z. L. Jiang, "A traceable outsourcing cp-abe scheme with attribute revocation," in *Trustcom/BigDataSE/ICESS, 2017 IEEE*, pp. 363–370, IEEE, 2017.

[19] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

[20] B. Lynn, "The stanford pairing based crypto library," *Privacy Preservation Scheme for Multicast Communications in Smart Buildings of the Smart Grid*, vol. 324, 2013.

[21] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 463–474, ACM, 2013.