IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 31, 2016, accepted November 18, 2016, date of publication November 29, 2016,
date of current version January 4, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2632132

Owner Specified Excessive Access Control for Attribute Based Encryption

FAWAD KHAN[1,2], HUI LI[2], (Member, IEEE), AND LIANGXUAN ZHANG[2]
[1]School of International Education, Xidian University, Xi'an 710071, China
[2]State Key Laboratory of Integrated Service Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China

Corresponding author: F. Khan (fawad.khan.xdu@gmail.com)

This work was supported by the National Natural Science Foundation of China under Grant U1401251 and Grant 61272457.

ABSTRACT Attribute-based encryption (ABE) has emerged as a promising solution for access control to diverse set of users in cloud computing systems. Policy can just specify whether (or not) any specific user should be given access to data, but it lacks to provide data owner the privilege to specify (how much) fraction, or (which) specific chunk from that data to be accessed or decrypted. In this paper, we address this issue, and propose a scheme that will give data owner excessive access control, so that he can specify specific chunk out of total data to be accessed by user depending on his attributes. In our scheme, a data owner can encrypt data over attributes specified in a policy, but even if user's attributes satisfy the policy; he can decrypt data (partially or fully) fractionally based on his attributes specified by owner. The owner can also prioritize user's access based on his designation, or hierarchal role in a specific organization. We also address to resolve the issue of attributes repetition, due to which the cost of computations in encryption by owner and ciphertext size is reduced. Furthermore, we achieve it with a single ciphertext over policy for entire data, and proof our scheme to be secure in the generic group and random oracle model. Theoretical comparisons of computations with existing constructions, and performance of the scheme evaluated in the Charm simulator is reasonable enough to be adopted in practice.

INDEX TERMS Attribute, partial, full, encryption, decryption, symmetric key, chunk, excessive access, repetition, fractional.

I. INTRODUCTION

Attribute Based Encryption (ABE) has evolved as an access control mechanism for large target community. Cloud storage is a service of cloud computing [30], utilized by data owners to outsource their data to the servers. ABE is considered as a promising solution for data access in cloud computing.

As cloud servers are not trust worthy, so owner undertakes the responsibility to encrypt its data before outsourcing it to the server. Owner defines an access policy for specific attributes that it wishes to be mandatory for data access, and then sends it along with ciphertext to the server. If attributes of user key satisfy the policy specified in ciphertext; user then can decrypt ciphertext correctly to get data. In ABE, for example, the University can share an examination notice, or provide access to particular data placed on its server (where professors are entitled to view all data while students have limited access to data) using policy named *Simple* defined as: University $\wedge$ (Professor *OR* Student), for users with attributes "University and Professor" , "University and Student" to retrieve the data. We tag this policy as *Simple*, because in rest of paper we will refer to this specific policy using its tag. Policy can just define here that any user with attributes can access the data, but it fails to define that out of those user attributes, which attribute's allows access to which fraction (chunk) of data. We further elaborate it by the scenario; let's consider a video provider is encrypting a video for attributes $Free_{User}$, and $Paid_{User}$. If the policy is defined to be: ($Free_{User}$ *OR* $Paid_{User}$), then individual users having attributes either $Free_{User}$, or $Paid_{User}$ can decrypt the video. Although, user with any (one) of these attributes will have access to data, but now the owner wants to restrict the user with $Free_{User}$ attribute to decrypt and view just starting "five" minutes of video, and user with $Paid_{User}$ attribute to decrypt and view the whole video. Although, both attribute's satisfy the policy and have access to data, but now they differ in the data being accessed using them. In other words, video provider in this case has restricted the access of $Free_{User}$ to a fraction of video instead of whole video.

This is an important issue, which should be addressed to give the data owner more privileges, so that he can specify how much (fraction) of data can be accessed by users using their attributes. To differentiate, there are two thing's; one is user's attributes that qualify it for data access, if they satisfy the policy specified by owner, while the other thing is how much fraction of data can be accessed by using those attributes. For *Simple* policy, as Student and Professor have access to dissimilar variant of data; hence, we list them under different attribute set W_i in policy, where W_i is a set of attributes in policy providing access to different variants of data. Another issue, to be addressed is the repeated attribute (University in Simple policy) appearing in multiple attribute sets W_i leading to more computation cost in encryption operation at data owner.

Although, for both the scenario's (*Simple* and Video Provider policy); which we explained above can be dealt with by defining two separate policies over ciphertexts (based on attribute-sets W_i), with different symmetric decryption keys for fractional data access; we argue that it should be done with a single ciphertext over one policy with no attributes repetition. Throughout the paper, we will use words fractional, variant, and partial interchangeably for referring to a chunk of data.

A. THIS WORK

In this paper, we address the issue of providing more access control to data owner, and also propose a technique to resolve an issue of attributes repetition in Linear Secret Sharing Scheme (LSSS) access matrix, which arise due to provision of excessive access. Removal of repeated attributes from access matrix leads to reduced computations in encryption operation and ciphertext size. Our proposed scheme, will allow owner to encrypt data under one policy and single ciphertext over attributes, and can specify (how much) fraction of data can be accessed by user using his individual attributes or combination of his attributes. User on retrieving the ciphertext will check that if his attributes satisfy the policy specified in ciphertext, then he can get access to either fraction-of or full data depending on his attributes as specified by owner in ciphertext. Moreover, owner can prioritize user's data access based on his hierarchal role in an organization. In this work, the conformation of user's attributes to satisfy ciphertext policy does not lead to entire data access, but a part of it; unless specified by owner.

B. PAPER ORGANIZATION

The rest of this paper is organized as follows. Related work is detailed in Section II, followed by Section III containing a brief overview of access structures, linear secret sharing scheme and multi-authority ABE. Section IV details our proposed scheme, while section V gives the performance analysis and security proof of our scheme. Section VI concludes the paper.

II. RELATED WORK

In this section, we will review attribute based encryption, access hierarchies, and chunk based encryption.

Attribute based encryption [2]–[4], [11], [12], [35], [36] has emerged as a valuable tool for access control in cloud computing systems. ABE find applications in places, where there are multiple sets of users in need to access different pieces of data. In order to comply with it, owner encrypts data over a ciphertext embedded with a policy of attributes; defining which individual attribute's or their combination can help decrypt data from ciphertext. There are two types of ABE. The first one named as Key-Policy ABE (KP-ABE). The first construction of KP-ABE was made by Goyal *et al.* [11]. In KP-ABE, the ciphertext is associated to attributes while the user decryption key is linked to access policy. To correctly decrypt, the policy in user key must conform to the attributes in ciphertext. The second type of ABE is Ciphertext-Policy ABE (CP-ABE). In CP-ABE [2], ciphertext is related to policy while attributes correspond to user decryption key. User can decrypt properly to get data if attributes in user key satisfy the policy in ciphertext. In single authority ABE schemes, a single certification authority is responsible for managing attributes, key generation and key distribution to users. This poses issues like Quality of Service (QoS) and scalability. To cater these problems, the foremost multi-authority ABE was proposed by Chase and Melissa [8], but the scheme required a central authority for coordination between all authorities. Lekwo *et al.* [4] then proposed the first decentralized multi-authority ABE in which all the authorities were independent from each other and hence required no coordination. Several other works [13]–[17], [34] were proposed to cater different issues regarding ABE but still the issue of giving more privileges to data owner, so that it can restrict user to (full or partial) fractional access of data based on its attributes is un addressed.

To give different access rights to users; the data cannot be encrypted with a single symmetric encryption key, but rather with different chunk keys. The idea of chunk based encryption [18], [20], [26], [27] is not new. In chunk based encryption, data is divided into several chunks and each one is encrypted using a different symmetric key. In [18], it is utilized for providing confidentiality and integrity to data in a cloud computing system while Yang *et al.* [20], utilized it for revocable data access control.

Access hierarchies [19], [22], [25] play an important role in granting access to users based on their roles. In this approach, several chunk keys are derived from the parent node key with the condition that chunk keys are derived only in the top-down manner, i.e., from the parent nodes to its descendant child nodes. To grant hierarchal access to users; keys are given to users in hierarchal order, i.e., the user having parent node key will have access to all data; in contrast the user having key with no descendant node has the access to only a part of data. A good construction of this is proposed by Atallahs *et al.* [22], which has been proven secure

against collusion. Vimercati *et al.* [19] proposed multi key encryption approach to give different access rights to users by adopting the scheme proposed in [22]. Several works in which the concept is utilized are: enforcing access for users on published XML documents [21], various variants of video quality provision [23], encrypted database access control [24], and cloud storage framework [29].

III. PRELIMINARY BACKGROUND

In the section, we will briefly discuss about access structures, Linear Secret Sharing Scheme (LSSS) and multi authority CP-ABE.

A. ACCESS STRUCTURES

Definition 3.1: Access Structure [7]. We let $P_1, \ldots, P_n$ represent a set of parties. A collection $X \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall\ Y, Z$: if $Y \in X$ and $Y \subseteq Z$, then $Z \in X$. Access structure specifically monotone will be a collection of non-empty subsets of $P_1, \ldots, P_n$. The sets in X are called authorized sets.

In our work, we let attributes similar to parties and we consider only monotone access structures.

B. LINEAR SECRET SHARING SCHEME

Our construction utilizes linear secret sharing scheme. We adapt the definition from [7].

Definition 3.2: Secret sharing scheme $\prod$ over a set of parties P is linear if:

1. Combination of shares from all parties forms a vector over Z_p.

2. A share generating matrix $\boldsymbol{A}$ exists for $\prod$. Matrix $\boldsymbol{A}$ has m rows and n columns. For $x = 1$ to m each x^{th} row of $\boldsymbol{A}$ corresponds to a party $P(x)$. We let a column vector $\boldsymbol{v} = \{s, v_2, \ldots, v_n\}$ be a sharing vector; where s is the secret to be shared is selected from Z_p, and $v_2, \ldots, v_n \in Z_p$. Then $\boldsymbol{A.v}$ is the vector of m shares of s according to $\prod$. The share $\lambda_i = (\boldsymbol{A}\ \overrightarrow{\boldsymbol{v}})_{\boldsymbol{i}}$ belongs to party $p(i)$.

We let S denote the set of attributes. Define $L \subseteq \{1, \ldots, m\}$ as $L = \{x | p(x) \in S\}$. There exists a vector (1,0,...,0) in the span of $\boldsymbol{A_x}$ indexed by L, where $\boldsymbol{A_x}$ represent rows of $\boldsymbol{A}$. For linear reconstruction, we have constants of the form $\{\ \boldsymbol{w_x} \in Z_p\}_{x \in L}$ such that, if λ_x are valid shares of secret s according to $\prod$, then s can be reconstructed by $\sum_{x \in L} \boldsymbol{w_x}\ \lambda_x = s$.

C. MULTI-AUTHORITY EAC CP-ABE

We briefly describe the algorithms that are part of the multi authority Excessive Access Control (EAC) CP-ABE scheme.

Global Setup$(\lambda) \rightarrow GP$: The algorithm takes as input the security parameter λ and outputs the global parameters GP for the system.

Authority Setup$(GP) \rightarrow SK, PK$: In this algorithm, the input is global parameters of the system and output corresponds to secret SK / public PK keys of the authorities.

Encrypt$(S, (A, \rho), GP, PK) \rightarrow CT$: Encrypt algorithm takes as input the message S (symmetric keys for data chunks), access matrix $(\boldsymbol{A}, \boldsymbol{\rho})$, global parameters and the public keys of authorities to output the ciphertext CT.

KeyGen $(GID, GP, i, SK) \rightarrow K_{i,GID}$: KeyGen algorithm takes as input the user identity GID, specific attribute i, GP and secret key of corresponding authority to generate a decryption key $K_{i,GID}$ for user.

Decrypt $(CT, GP, K_{i,GID}) \rightarrow S$: To decrypt the ciphertext, this algorithm takes as input CT, GP and the set of user attribute keys. For successful decryption the user attribute keys should correctly satisfy the access matrix in the ciphertext.

Definition 3.3: A multi authority EAC CP-ABE scheme is correct if the GP is obtained from global setup algorithm, CT from encrypt algorithm, keys $K_{i,GID}$ corresponding to specific attributes of user GID are generated using keygen algorithm and message (S) is obtained from CT using decrypt algorithm, if the set of attributes in key satisfy the access matrix in CT.

IV. OUR CONSTRUCTION

In this section, we will provide a detailed construction of our scheme.

A. SECURITY MODEL

We define the security game for EAC CP-ABE system between challenger and attacker in the following way. We assume that the adversary can corrupt authorities statically but can make queries adaptively till the end of game. This same model has been used by Chase *et al.* [9], and Chase and Melissa [8] respectively. In our model, we give more powers to attacker, that he can choose public keys of corrupt authorities by himself; instead given to him by challenger at the start of game.

We let S denote the set of all the authorities in system and U denote the attributes universe. Each attribute belongs to one authority.

Setup: The global setup algorithm is run. The attacker specifies a set of corrupt ($S_0 \subset S$) authorities to the challenger. Challenger then obtains the public and private keys of uncorrupt ($S - S_0$) authorities by running the authority setup algorithm. Finally, challenger reveals the acquired public keys to the attacker.

Phase 1: The attacker queries challenger for key pairs (i, GID) corresponding to attribute i of good authority and user identity GID. The challenger replies to the queries by sending out the key pairs of the form $K_{i,GID}$ to the attacker. For attributes in corrupt authority, the attacker can generate decryption keys by himself.

Challenge Phase: Attacker then specifies two equal length messages under the access structure $(\boldsymbol{A}, \boldsymbol{\rho})$. Moreover, specifies for each attribute in ρ that it belongs to which attribute set W_i. Let V denote the subset of rows of access matrix A controlled by corrupt authorities. We denote V_{GID} as the subset of rows of A for which the attacker can acquire the keys (i, GID) corresponding to attribute i and identity GID

for uncorrupt authorities. The constraint on access matrix is that the subspace spanned by VUV_{GID} should not include (1,0,...,0) in its span for any of attribute set W_i. In other words, the attacker cannot ask for those specific keys which he can combine with keys of corrupt authorities to allow successful decryption for specific attribute and identity *GID*. Also the attacker gives public keys of corrupt authorities attributes to challenger which appears in the image ρ. Challenger flips a random coin $\beta = \{0, 1\}$ and encrypts a message M_β according to access policy.

Phase 2: Attacker makes further key queries (i, GID) but under the constraint that queries dont violate the challenge matrix $(\boldsymbol{A}, \boldsymbol{\rho})$.

Guess: Attacker submits a guess β' for β. Attacker wins the game if $\beta' = \beta$. The advantage of attacker in the security game is $\Pr[\beta' = \beta] - 1/2$.

Definition 4.1: A multi authority ciphertext policy attribute based encryption scheme with excessive access control is secure (against static corruption of authorities) if all polynomial time adversaries have at most a negligible advantage in security game.

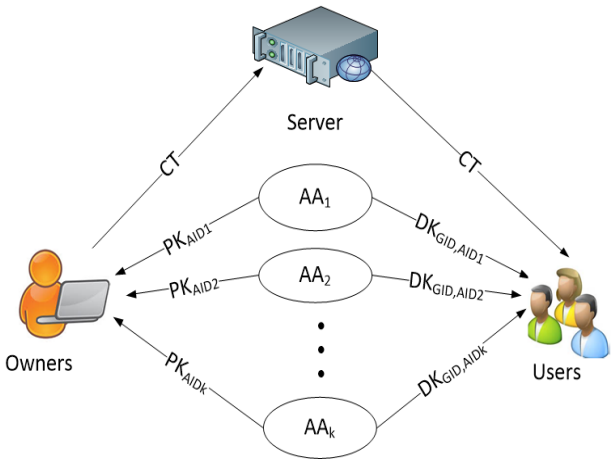

FIGURE 1. System Model.

B. SYSTEM MODEL

We let a multi-authority cloud storage system as shown in FIGURE1. It consists of owners, users, server, and attribute authorities (AA).

Owner An entity who wants to publish data on the server to be retrieved later by other users based on their attributes. For publishing data it will first split the data into chunks and encrypt each chunk with a new symmetric key followed by defining a decryption policy based on attributes.

Server

It is an entity used for outsourcing owner's data and providing data access to users.

User It can request ciphertext from server, but can decrypt only when his decryption keys satisfy the access policy in ciphertext.

Attribute Authority Each Attribute Authority (AA) generates a public, private key pair for itself. Moreover, it undertakes the responsibility for managing attributes and generating keys for users based on their attributes handled by AA. Authorities work in a decentralized fashion; hence no coordination is required between them.

$E_{K_1}(m_1)$	$E_{K_2}(m_2)$	...	$E_{K_n}(m_n)$

FIGURE 2. Data Chunks Encryption.

C. IDEA OF OUR CONSTRUCTION

In many situations, owner does not want to give access of entire data, but a part or fraction of it. To provide owner more privileges, so that he can specify how much fraction of data can be accessed by user based on his attributes; data needs to be divided into chunks, and encrypted using different symmetric keys as shown in FIGURE2, where m_1 is data chunk 1 encrypted with symmetric key K_1, m_2 is data chunk 2 encrypted with symmetric key K_2 and so on for other data chunks.

To enforce, fractional or partial access of data, one challenge is that owner must encrypt data using only one policy and ciphertext for all individual data chunks encrypted with different symmetric keys.

We define a scenario to illustrate our idea. Let's consider a research being carried out by university for designing disease measuring equipment for a hospital in collaboration with some company. People from three different domains hospital, university, and company are involved in work and they have access to different fractions of data based on their attributes. We define an excessive access control policy named *Complex* as: {(University $\wedge$ Student) *OR* (Hospital $\wedge$ Patient)} *OR* {(University $\wedge$ Professor) *OR* (Hospital $\wedge$ Nurse)} *OR* {Company $\wedge$ (HR *OR* Marketing *OR* Finance)} *OR* {(Hospital $\wedge$ Doctor) *OR* (Company $\wedge$ Boss)}. The first glance of this *Complex* policy shows that any individual with specific set of attributes will have provision to same and all data, but here owner wants to give different access rights to users.

We use this {} notation to indicate that attributes enclosed in these refer to an attribute set W_i. Each W_i contains a diverse set of users having access to same data chunk. With in an attribute set W_i, if $\sum c_x \lambda_x = q_i$; then user can have access to data. For *Complex* policy we have four attribute sets as: (1). W_1 = {(University $\wedge$ Student) *OR* (Hospital $\wedge$ Patient)}, (2). W_2 = {(University $\wedge$ Professor) *OR* (Hospital $\wedge$ Nurse)}, (3). W_3 = {Company $\wedge$ (HR *OR* Marketing *OR* Finance)}, and (4). W_4 = {(Hospital $\wedge$ Doctor) *OR* (Company $\wedge$ Boss)}; where W_i is an attribute set. FIGURE3 shows the data access for each attribute set W_i. The point worth noting here is that attribute's "University, Hospital, and Company" appear in more than one attribute sets because user's besides their attribute's need these to access data. Like in this policy, "Student" belonging to W_1 and "Professor" from W_2 both needs "University" attribute to access their data variant. Excessive control can be enforced in this case, via partitioning the data into three parts and encrypting chunks with

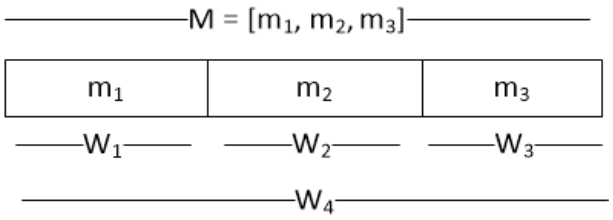

FIGURE 3. Fractional Data Access of Attributes Sets.

three different symmetric keys. Next, for enforcing excessive data access, we compare and contrast the two approaches mentioned below.

Basic Scheme In, Traditional-ABE (T-ABE) schemes, all users have access to one and only single data encrypted using a symmetric key enclosed in M. Moreover, this M is encrypted using a secret s raised to a pairing in the form of $M * e(g, g)^s$. The secret s is shared among user's attributes in ciphertext using LSSS matrix shares λ_x. Using T-ABE schemes, in-order for owner to give different data access to users for *Complex* policy; it will have to break the policy into 4 individual attribute sets W_i policies and then compute the ciphertext for each of the policy. This is due to the fact that for T-ABE schemes we cannot have more than one secret s in a ciphertext for policy. As mentioned earlier, three attributes "University, Hospital and Company" repeats in attribute sets W_i, or in other words these are shared by other attributes to access data. For *Complex* policy, there are overall 12 "distinct" attributes, but due to repetition the overall number is 16. This repetition increases the computational effort while computing ciphertexts in encryption operation by owner and moreover, elongates its size.

Our Scheme We propose our method, in which the owner can enjoy the privilege of excessive access control by defining a single policy over one ciphertext with no attributes repetition. Moreover, users having access to various different chunks of data will all have to satisfy that just one policy. To do so, the excessive access *Complex* policy should be rewritten with no-attributes repetition for our scheme in its compact form as: (University $\wedge$ (Professor *OR* Student)) *OR* (Hospital $\wedge$ (Patient *OR* Doctor *OR* Nurse)) *OR* (Company $\wedge$ (HR *OR* Marketing *OR* Finance *OR* Boss)). The LSSS matrix for the compact complex policy is also constructed. In our approach, instead of defining separate policies like in traditional ABE schemes; we define different secrets q_i to be shared for attribute sets W_i. The sharing vectors for secrets q_i are evaluated by the formed LSSS matrix with distinct attributes, but due to repetition of attribute's between multiple W_i; shares λ_x for repeated attributes are also multiple. For *Complex* policy, we have 16 evaluated λ_x shares for attributes belonging to four 4 attribute sets. We remove the repeated attributes λ_x shares, so that they correspond exactly to 12 "distinct" attributes, and also conform to secrets q_i by employing our proposed methodology mentioned in section IV - D of this paper.

D. ELIMINATING REPEATED ATTRIBUTES

Based on the fact that summation of attribute's share's λ_x leads to secret reconstruction, i.e., $\sum c_x \lambda_x = q_i$ in W_i if the attributes $\in W_i$ form a span of (1,0,. . .0). We exploit this secret re-construction property to eliminate the repeated attributes shares appearing across multiple W_i. We have two types of attributes namely: O_{attr} and R_{attr}.

O_{attr}, represents other attribute appearing with repeated one in policy, c_O denotes coefficient of other attribute share, R_{attr} for repeated attribute appearing multiple times, and c_R is the coefficient of repeated attribute share. c_O, and c_R correspond to values c_x for which $\sum c_x \lambda_x = q_i$ (secret reconstruction). For *Simple* policy, R_{attr} corresponds to University while O_{attr} represents Student and Professor. For eliminating the repeated attributes we utilize the following equation:

$$(O_{attr}) = (1/c_O)\{(q_i) - (c_R)(R_{attr})\}$$

Where q_i denotes secret to be shared for an attribute set W_i. Elimination is done by putting all values in equation, and by fixing the value of repeated attribute R_{attr} share; there by obtaining the values for other attributes O_{attr} shares. We refer reader to Appendix A, for an example of using this method for eliminating the repeated attributes.

E. PROPOSED SCHEME

In our construction, policy is defined over the ciphertext only once for all data chunks with no attribute repetition by employing the methodology as stated above.

Global Setup$(\lambda) \rightarrow GP$: In global setup, a bilinear group G of prime order p is chosen. Global parameters are set to p, g, $e(g, g)$ and H; where g is a generator of group G and H is a hash function that maps global identities GID to elements in G. We will model H as random oracle for security proof.

Authority Setup$(GP) \rightarrow SK, PK$: Each authority selects for itself a random value $t \in Z_p$. For each attribute i that belongs to authority, it chooses a random value $\alpha_i \in Z_p$. It keeps values $\{t, \alpha_i \forall i\}$ as secret key, SK and publishes $\{g^t, e(g, g)^{\alpha_i} \forall i\}$ as public key, PK.

Encrypt$(S, (A, \rho), GP, PK) \rightarrow CT$: The owner first takes data $M = [m_1, m_2, \ldots, m_k]$ encrypted with $S = [S_1, S_2, \ldots, S_k]$, where S_i corresponds to the chunk m_i and S corresponds to M. After that, he defines an excessive access policy $W = [W_1\ OR\ W_2\ OR ... OR\ W_k\ OR\ W_{k+1}]$, where W_i corresponds to symmetric chunk key S_i, and W_{k+1} corresponds to symmetric key S. Then, he rewrites the policy without attributes repetition and forms an LSSS matrix $\boldsymbol{A}$ of size m x n. The algorithm takes as input; a message S (symmetric keys for data chunks), global parameters, an access matrix $\boldsymbol{A}$ of size m x n with $\boldsymbol{\rho}$ containing a map of its rows to attributes, and PK's from relevant authorities. To proceed, it will first choose random values $q_i, v_{i,2}, \ldots, v_{i,n} \in Z_p$ to form a sharing vector $\boldsymbol{v_i} = \{q_i, v_{i,2}, \ldots, v_{i,n}\}$; where q_i is secret to be shared for each attribute set W_i. Further, it computes $\lambda_x = \boldsymbol{A_x} \cdot \boldsymbol{v_i}$ for $\boldsymbol{A_x} \in W_i$ where $\boldsymbol{A_x}$ is x^{th} row of $\boldsymbol{A}$. Moreover, it will choose a random vector $\boldsymbol{w} \in Z_p$ of length n with 0 as its first entry. Compute $w_x = \boldsymbol{A_x} \cdot \boldsymbol{w}$.

Then it will use the algorithm in section D to remove the repeated values of shares λ_x, so that attributes are not repeated, and finally it computes the ciphertext as:

$$\text{CT} = \{C_i = S_i \cdot e(g,g)^{q_i}, C_{1,x} = e(g,g)^{\lambda_x} \cdot e(g,g)^{\alpha_{\rho(x)} w_x}, \\ C_{2,x} = g^{t w_x} \text{ for } \rho(x) \in W_i | i = 1, 2, \ldots, k+1\}.$$

The owner then sends CT, $(\boldsymbol{A}, \boldsymbol{\rho})$ for compact policy along with excessive access policy $W = [W_1\ OR, \ldots, OR\ W_{k+1}]$ to the server. Ciphertext C_i values correspond to attribute sets W_i in access policy.

KeyGen $(GID, GP, i, SK) \rightarrow K_{i,GID}$: To create a key for user GID corresponding to an attribute i of authority, it computes $K_{i,GID} = g^{\alpha_i/t} \cdot H(GID)^{1/t}$.

Decrypt $(CT, GP, \{K_{i,GID}\}) \rightarrow S$: For decryption, the primary assumption is that the ciphertext is encrypted under access matrix $(\boldsymbol{A}, \boldsymbol{\rho})$. The user will determine his attributes belonging to particular attribute set using excessive access policy W, the span $(1, 0, \ldots, 0)$ and exact ciphtertext corresponding to his keys using $(\boldsymbol{A}, \boldsymbol{\rho})$. If the decryption user keys $\{K_{\rho(x),GID}\}$ can form a span $(1, 0, \ldots, 0)$ over the subset of access matrix rows $\boldsymbol{A_x} \in W_i$, then user will choose constants $u_x \in Z_p$, so that $\sum_x u_x \boldsymbol{A_x} = (1, 0, \ldots, 0)$ where $\boldsymbol{A_x} \in W_i$, and then will decrypt as follows: $\prod_x (C_{1,x}/e(C_{2,x}, K_{\rho(x),GID}))^{u_x} = e(g,g)^{q_i}$.

After correctly finding $e(g,g)^{q_i}$ user will divide this by value of C_i (corresponding to his attribute set W_i) to obtain S_i which is symmetric chunk key for m_i.

F. CORRECTNESS

The proposed excessive access control scheme is correct. To decrypt, the user will first choose constants $u_x \in Z_p$, so that $\sum_x u_x \boldsymbol{A_x} = (1, 0, \ldots, 0)$; then will decrypt as follows:

$$\begin{aligned}
&\prod_x (C_{1,x}/e(C_{2,x}, K_{\rho(x),GID}))^{u_x} \\
&= \prod_x (e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} w_x} / e(g^{t w_x}, g^{\alpha_{\rho(x)}/t} \cdot H(GID)^{1/t}))^{u_x} \\
&= \prod_x (e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} w_x} / e(g,g)^{\alpha_{\rho(x)} w_x} \\
&\quad \cdot e(g, H(GID))^{w_x})^{u_x} \\
&= \prod_x (e(g,g)^{\lambda_x} / e(g, H(GID))^{w_x})^{u_x} \\
&= e(g,g)^{q_i}
\end{aligned}$$

V. ANALYSIS OF THE SCHEME

In this section, we will analyze the performance of our scheme, and give a security proof for our construction.

A. PERFORMANCE EVALUATION

We will first compare the theoretical computations and security model of our scheme with existing Water's constructions BSW07 [2], W09 [3], LBW11 [4], RW13 [5], and RW15 [6], and then we will demonstrate with the help of two policies the effect of attributes repetition on the performance of schemes. Finally, we will present our scheme running time performance.

TABLE 1. Comparison with existing water's constructions.

Sch	SM	$KeyGen$	$Encrypt$	$Decrypt$
[2]	FS	$(2m+2)E$	$(2n+2)E$	$(2z+1)P + 2zE$
[3]	SS	$(m+2)E$	$(3n+2)E$	$(2z+1)P + zE$
[4]	FS	$(2m)E$	$(5n+1)E$	$(2z)P + zE$
[5]	SS	$(3m+4)E$	$(5n+2)E$	$(3z+1)P + zE$
[6]	FS	$(l+2m)E$	$(6n+1)E + P$	$(3z)P + zE$
Our's	FS	$(2l+m)E$	$(3n+W)E$	$zP + zE$

In TABLE I, the notations used for various entities are: SM, SS, FS for security model, selective and full security, l for the number of authorities, m for number of user attributes, n for number of attributes in access structure, W for attribute sets in access structure, and z for number of user's attributes utilized for satisfying the policy. Moreover, we represent E for exponential and P for pairing operation. TABLE I contains the operations carried out by various constructions in key generation, encryption and decryption operations. As seen from the table our scheme has comparable computations in terms of key generation and encryption but it surpasses all other schemes in decryption by utilizing just one pairing and exponential operation per attribute.

TABLE 2. Comparison of computations for simple and complex policy.

$Policy$	$Scheme$	$KeyGen$	$Encrypt$	$Decrypt$
$Simple$	[2]	$12E$	$12E$	$10P + 8E$
	[3]	$8E$	$16E$	$10P + 4E$
	[4]	$8E$	$22E$	$8P + 4E$
	[5]	$20E$	$24E$	$14P + 4E$
	[6]	$12E$	$26E + 2P$	$12P + 4E$
	Our's	$12E$	$11E$	$4P + 4E$
$Complex$	[2]	$40E$	$40E$	$45P + 36E$
	[3]	$24E$	$56E$	$45P + 18E$
	[4]	$32E$	$84E$	$36P + 18E$
	[5]	$64E$	$88E$	$63P + 18E$
	[6]	$40E$	$100E + 4P$	$54P + 18E$
	Our's	$32E$	$40E$	$18P + 18E$

Now, we will demonstrate the effect of attributes repetition on the performance of schemes. For traditional ABE schemes, in-order to provide excessive access; the owner needs to divide the policy based on the attributes sets W_i, and then encrypt separate ciphertexts for those policies due to which attributes shared with in multiple W_i are repeated. In our scheme, attributes are not repeated; hence the ciphertext size is short, and fewer computations have to be performed in encryption operation by owner. Moreover, we achieve it with a single ciphertext over one policy. TABLE II lists computations performed by the all schemes for *Simple* and *Complex* policy. Our scheme has less computational operations for both the policies in encryption and decryption algorithms.

We have implemented our scheme in Charm [31], [33], which is a cryptographic tool for defining and evaluating pairing based constructions. It is based on a high level language python and utilizing [28], pairing based cryptography libraries. All charm routines work on underlying asymmetric groups, even if the basic constructions are in

symmetric groups. So, we need to convert our scheme to its asymmetric version although it reduces efficiency; now we have three multiplicative cyclic groups G_1, G_2 and G_T of prime order p, and the bilinear map $e : G_1 \text{ x } G_2 \rightarrow G_T$ is between them. All our simulations are executed on a (3 GB allocated Ram) Hyper-V Virtual Machine running (Ubuntu 14.04, Python3.4.3 and Charm-Crypto-0.43) on dell inspiron laptop Intel(R) Core(TM) i5-3337U CPU@ 1.80GHz with 8 GB Ram.

TABLE 3. Average running time in milliseconds(*ms*) of our scheme for (*S*), (*C*) Policy with parameters.

P	GS	AS	KG	Enc	Dec	$Attr$	U	V
S	10.12	12.19	31.78	20.97	9.84	3	2	2
C	10.85	52.87	146.64	69.20	74.9	12	9	4

In TABLE III, we give the running time (ms) of global setup: GS, authority setup: AS, key generation: KG, encryption: Enc, decryption: Dec algorithms for our scheme evaluated in Charm for *Simple* and *Complex* policies. The notations $Attr$, U and V represent number of attributes, users and data variants for policy P in TABLE III. In prior one, there are two (2) user's having access to different variants of data, while in *Complex* policy there are nine (9) user's having access to four (4) variants of data. Decryption time evaluated for *Simple* policy includes time of both Student and Professor to get their share of symmetric key. Similarly it applies for *Complex* policy containing nine users from four different attribute sets.

B. SECURITY PROOF

We proof our scheme to be secure using generic bilinear group model previously employed in [1], [2], [4], and [10]. We will model H (hash function) as a random oracle. Security model assures that adversary cannot succeed to break our scheme given only black box access to group operations and H.

We describe the generic bilinear group model as described in [1]. We let ψ_0 and ψ_1 are two random encodings of additive group Z_p. Each of ψ_0 and ψ_1 is an injective map from Z_p to $\{0, 1\}^m$ for $m > 3log(p)$. Formally, we represent the groups as: $G_0 = \{\psi_0(x) : x \in Z_p\}$ and $G_1 = \{\psi_1(x) : x \in Z_p\}$. Assume that we have access to oracles for evaluating the group operations in G_0 and G_1. Moreover, we have the oracle to compute the non-degenerate bilinear map e: $G_0 \times G_0 \rightarrow G_1$.

In the security game, the attacker has to distinguish between $C_i = M_0 e(g, g)^{q_i}$ and $C_i = M_1 e(g, g)^{q_i}$. We now consider a modification in the game [2], where the attacker must distinguish between $C_i = M_0 e(g, g)^{q_i}$ and $C_i = M_0 e(g, g)^{a_i}$, where $a_i \in Z_p$ is selected for each attribute set W_i. We simplify the notations that we use as: g denote $\psi_0(1)$, g^x denote $\psi_0(x)$, $e(g, g)$ denote $\psi_1(1)$ and $e(g, g)^y$ denote $\psi_1(y)$.

We now simulate the modified security game in generic bilinear group model, where C_i is set to $e(g, g)^{a_i}$. Moreover, S represents the set of authorities and U represents the set of attributes universe. Simulator runs the global setup algorithm and gives g to the attacker. Attacker then specifies a set $S' \subset S$ of corrupt authorities, and discloses it to the simulator. Simulator randomly chooses $t \in Z_p$ for uncorrupted authorities, and $\alpha_i \in Z_p$ where $i \in U$ corresponds to attributes that are controlled by uncorrupted authorities; queries group oracles for evaluating g^t, $e(g, g)^{\alpha_i}$ and gives these values to attacker.

Attacker then requests $H(GID)$ for the first time. Simulator chooses a random value $h_{GID} \in Z_p$, queries group oracles for $g^{h_{GID}}$ and sends it to attacker. Also, the simulator keeps a copy of the sent value, so that the requested GID value in future will be dealt with the same evaluated value. Attacker then requests a key $K_{i,GID}$ for an attribute i belonging to a particular authority and identity GID. In response, simulator computes $g^{\alpha_i/t} \cdot H(GID)^{1/t}$ by querying the group oracles, and send it back to the attacker. After some time, attacker will specify an access matrix $(\boldsymbol{A}, \boldsymbol{\rho})$ for challenge ciphertext with attributes specified for attributes sets W_i. Moreover, values of corrupt authority attributes that appear in ρ (corresponding to rows of $\boldsymbol{A}$) of access matrix will be sent by attacker to simulator. Simulator confirms the validity of these attributes by querying group oracles.

Simulator will now produce the challenge ciphertext. To follow up, it will first choose random values $q_i, v_{i,2}, \ldots, v_{i,n} \in Z_p$ to form a sharing vector $\boldsymbol{v_i} = \{q_i, v_{i,2}, \ldots, v_{i,n}\}$; where q_i is secret to be shared for each attribute set W_i. Further, it computes $\lambda_x = \boldsymbol{A_x} \cdot \boldsymbol{v_i}$ for $\boldsymbol{A_x} \in W_i$ where $\boldsymbol{A_x}$ is x^{th} row of LSSS matrix $\boldsymbol{A}$ with no repeated attributes. Moreover, selects a vector $\boldsymbol{w} = \{0, w_2, \ldots, w_n\}$ where each $w_2, \ldots, w_n$ is selected randomly from Z_p and evaluates $w_x = \boldsymbol{A_x} \cdot \boldsymbol{w}$. Simulator will give λ_x shares values to elimination algorithm for removal of repeated attributes. Finally it will select random values $a_i \in Z_p$ for W_i. With the help of group oracles the simulator now computes the ciphertext as:

$$\{C_0 = e(g, g)^{a_i}, C_{1,x} = e(g, g)^{\lambda_x} \cdot e(g, g)^{\alpha_{\rho(x)} w_x}, \; C_{2,x} = g^{t w_x} \; \forall \, x\}.$$

The challenge ciphertext is given to the attacker. We argue that by all, but with negligible probability, an attacker view regarding if C_i is set to $e(g, g)^{a_i}$ in place of $e(g, g)^{q_i}$ is identical in simulation. This illustrates that attacker cannot attain non negligible advantage in modified security game; hence, he cannot gain non negligible advantage in real security game.

We condition on attackers queries to input values, as the value's given to attacker during simulation, or the values which he received in response of previous queries which he made to oracles. The event occurs with greater probability. As ψ_0 and ψ_1 are random injective maps from Z_p into a set with greater than p^3 elements; to guess an element appearing in image of ψ_0, ψ_1 occurs with negligible probability which has not been attained before.

Under aforementioned condition, attacker can query as a multi variate polynomial in variables $a_i, \alpha_j, t, \boldsymbol{\gamma_x}, \boldsymbol{w_x}, h_{GID}$, where j stands for uncorrupted authorities, x ranges over rows of challenge access matrix and *GID* ranges over allowed identities. We take $\boldsymbol{\gamma_x}$ as the linear combination of variables $(q_i, v_{i,2}, \ldots, v_{i,n})$ for attribute sets W_i and $\boldsymbol{w_x} = (0, w_2, \ldots, w_n)$. Further, we state that for each different pair of queries responding to unlike polynomials, attacker receives different answers. Difference is non-zero for random assignment of values to variables for two query polynomials. This event occurs with greater probability which we can realize using union bound and Schwartz-Zippel lemma as the polynomials have at most degree 4.

We see that a_i only appears as $e(g, g)^{a_i}$, so the queries attacker can make about a_i will be of the form $ca_i +$ other terms, where c is constant. Attackers view can change only when it makes two different polynomial queries, f and f' into G_1 but if it replace $a_i = q_i$; the result will be same (one) polynomial. This implies that, $f - f' = ca_i - cq_i$ for some constant c. We conclude that attacker can query cq_i.

TABLE 4. Possible Query Terms.

α_i	t
h_{GID}	$\alpha_i/t + h_{GID}/t$
$\lambda_x + \alpha_{\rho(x)} w_x$	$t w_x$
$\alpha_i w_x + h_{GID} w_x$	h_{GID}/t
$h_{GID}\alpha_i/t + h_{GID} h_{GID}\prime/t$	$\alpha_i/t_j + h_{GID}/t_i t_j$
$h_{GID} h_{GID}\prime$	$(t w_x)(t\prime w_x\prime)$
$(\alpha_i/t_i + h_{GID}/t_i)(\alpha_j/t_j + h_{GID}\prime/t_j)$	$t_i t_j$

Now we will show that a query cq_i cannot be made by attacker, and hence we arrive at a contradiction. We can see the possible queries attacker can make in TABLE IV. By inspecting we came at a conclusion that attacker can only make queries of the form which are linear combinations of 1, a_i and other terms appearing in TABLE IV.

We remind that attacker knows the values of α_i, t for corrupted authorities; that's why the linear combinations of these values can appear in TABLE IV .

Recall that q_i can be constructed by $\lambda_x = \boldsymbol{A}_x \cdot \boldsymbol{v_i}$ where $\boldsymbol{v_i} = (q_i, v_{i,2}, \ldots, v_{i,n})$ for attribute set W_i. Hence the only appearance of q_i in TABLE IV can be constructed using the linear combination of λ_x. To order query of the form cq_i; attacker needs to choose constants β_x such that $\sum_x \lambda_x = cq_i$ by asking for query $(\lambda_x + \alpha_{\rho(x)} w_x)$ to form $\beta_x(\lambda_x + \alpha_{\rho(x)} w_x)$. For corrupt authorities attributes, attacker can construct polynomials of the form $-\beta_x \alpha_{\rho(x)} w_x$ to cancel out this term for the above polynomial. For uncorrupted authorities attributes, attacker needs to query $(\alpha_{\rho(x)} w_x + h_{GID} w_x)$ this; in-order to cancel out $-\beta_x \alpha_{\rho(x)} w_x$, which leaves an extra term of $-\beta_x h_{GID} w_x$. We note that attacker can access this term $(\alpha_{\rho(x)} w_x + h_{GID} w_x)$ if it requests for a key corresponding to a particular attribute $\rho(x)$ and identity *GID*.

The gathering of these terms for each identity *GID* will cancel this term only if the span (1,0,. . .,0) of length n vector is in the rows $\boldsymbol{A_x} \in \boldsymbol{W_i}$ of $\boldsymbol{A}$ belonging to corrupt authorities, or uncorrupted one's for which he acquired the keys for $(\rho(x), GID)$. Under this condition, the attacker has broken the rules of security game and requested for a set of keys for an identity *GID* with which he is capable to decrypt the challenge ciphertext.

Therefore, we have presented that attacker cannot construct a query of the form cq_i for some constant c. Hence, under these conditions that hold with all but with negligible probability, we state that the attackers view when a_i is random is identical to when $a_i = q_i$. This proves that the attacker cannot attain a non-negligible advantage in the security game.

VI. CONCLUSION

In this paper, we have proposed an excessive access control scheme for data owner using a single ciphertext over policy without attributes repetition. The owner can enjoy limiting the user's access to just specified chunks of data, instead of whole data. Moreover, owner can also grant privileged data access to users based on their hierarchal role in a specific organization. Comparison, in contrast to traditional approaches depicts its effectiveness in terms of providing variant data access in a single policy over ciphertext, and by less computations in encryption and decryption operation. Our proposed scheme is proven secure in generic group and random oracle model. Performance evaluation of scheme in Charm simulator is good to adopt it in practice. We will try to further enhance its performance in future, and extend it to other types of access structures besides LSSS.

APPENDIX
REMOVING REPEATED ATTRIBUTES FROM LSSS MATRIX

Here, we demonstrate how to remove the repeated attributes from LSSS matrix. Suppose the data owner wants to share fractional access of data using the "Simple" policy University $\wedge$ (Professor *OR* Student). For this policy, the attribute sets are $W_1 =$ (University $\wedge$ Professor) and $W_2 =$ (University $\wedge$ Student). Using T-ABE schemes, this policy is broken down based on the attribute sets W_i, and a separate ciphertext will be evaluated for both W_i. Attribute University appears in both W_i; hence, it will be evaluated for both the ciphertexts.

For *Our* proposed scheme, the data owner will write this policy in its compact form with no attributes repetition as: University $\wedge$ (Professor *OR* Student). For this compact policy the LSSS matrix $\boldsymbol{M}$ based on AND-OR gates [4], [32] is shown in FIGURE4 (a). For demonstration purpose (to have an idea regarding secret sharing and its re-construction), the shares of secret "s" are calculated as λ_1, λ_2 and λ_3. For re-construction the users based on their attributes will find the coefficients c_i by the relation $\sum c_i M_i =$(1,0,. . .,0). In this case c1 = c2 = c3 = 1. Combination of either Professor or Student share with University share will lead to re-construction of secret in FIGURE4 (a).

For excessive access control, we need to share 2 secrets q_1, q_2 accordingly for W_1, W_2. Hence, we encounter a problem that attribute "University" appears in each of the LSSS matrix as seen in FIGURE4 (b), (c) for W_1 and W_2. To solve this problem, we use our proposed approach mentioned in

$$\begin{matrix} University \\ Professor \\ Student \end{matrix}\begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} s=15 \\ 2 \end{bmatrix} = \begin{bmatrix} 17 \\ -2 \\ -2 \end{bmatrix}\begin{matrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{matrix} \quad (a)$$

$$\begin{matrix} University \\ Professor \end{matrix}\begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} q_1=7 \\ 10 \end{bmatrix} = \begin{bmatrix} 17 \\ -10 \end{bmatrix}\begin{matrix} \lambda_1 \\ \lambda_2 \end{matrix} \quad (b)$$

$$\begin{matrix} University \\ Student \end{matrix}\begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} q_2=11 \\ 4 \end{bmatrix} = \begin{bmatrix} 15 \\ -4 \end{bmatrix}\begin{matrix} \lambda_1 \\ \lambda_3 \end{matrix} \quad (c)$$

$$\begin{matrix} University \\ Professor \\ Student \end{matrix}\begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} q_1=7 \\ q_2=11 \end{bmatrix} = \begin{bmatrix} 10 \\ -3 \\ 1 \end{bmatrix}\begin{matrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{matrix} \quad (d)$$

FIGURE 4. LSSS Matrix (a) Shares of secret s for all attributes, (b). Shares of secret q_1 for attribute set W_1, (c). Shares of secret q_2 for attribute set W_2, (d). Shares of q_1&q_2 for all attributes with no repetition.

section IV - D of paper. The equations for determining the shares of Professor and Student are:

$$Professor, \lambda_2 = (1/c_2)\{(q_1) - (c_1)(\lambda_1)\}$$
$$Student, \lambda_3 = (1/c_3)\{(q_2) - (c_1)(\lambda_1)\}$$

Take any random value of $\lambda_1 \in Z_p$ (here its taken as 10), putting values of q_1, q_2, c_1, c_2 and c_3, we get values of λ_2, λ_3 as seen in FIGURE4 (d). We note here, that the users having attributes (either Professor, OR Student) if combine their shares λ_2, λ_3 with λ_1 corresponding to University; this will lead to one of the secret recovery either q_1 or q_2. Different secret reconstruction will lead to a variant data decryption key. The attribute shares in FIGURE4 (d) will be used for evaluating the ciphertext by owner. Finally, comparing FIGURE4 (a), (d) we see that in-contrast to a single secret being shared in prior one, we can share multiple secrets over same set of attributes and policy with no repetition.

REFERENCES

[1] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2005, pp. 440–456.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[3] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC*. Berlin, Germany: Springer, 2011, pp. 53–70.

[4] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2011, pp. 568–588.

[5] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 463–474.

[6] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2015, pp. 315–332.

[7] A. Beimel, "Secure schemes for secret sharing and key distribution," PhD thesis, Israel Inst. Technol., Technion, Haifa, Israel, 1996. [Online]. Available: https://www.cs.bgu.ac.il/ beimel/Papers/thesis.pdf

[8] M. Chase, "Multi-authority attribute based encryption," in *Theory of Cryptography*. Berlin, Germany: Springer, 2007, pp. 515–534.

[9] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 121–130.

[10] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 1997.

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

[12] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2010, pp. 62–91.

[13] J. Li, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Information Security*. Berlin, Germany: Springer, 2009, pp. 347–362.

[14] J. Lai, R. H. Deng, and Y. Li, "Fully secure cipertext-policy hiding CP-ABE," in *Information Security Practice and Experience*. Berlin, Germany: Springer, 2011, pp. 24–39.

[15] N. Doshi and D. Jinwala, "Hidden access structure ciphertext policy attribute based encryption with constant length ciphertext," in *Advanced Computing, Networking and Security*. Berlin, Germany: Springer, 2011, pp. 515–523.

[16] X. Li, "Efficient ciphertext-policy attribute based encryption with hidden policy," in *Internet and Distributed Computing Systems*. Berlin, Germany: Springer, 2012, pp. 146–159.

[17] Y. Zhang, "Anonymous attribute-based encryption supporting efficient decryption test," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 511–516.

[18] N. Virvilis, S. Dritsas, and D. Gritzalis, "A cloud provider-agnostic secure storage protocol," in *Critical Information Infrastructures Security*. Berlin, Germany: Springer, 2010, pp. 104–115.

[19] D. Vimercati *et al.*, "A data outsourcing architecture combining cryptography and access control," in *Proc. ACM Workshop Comput. Secur. Archit.*, 2007, pp. 63–69.

[20] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.

[21] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *Proc. 29th Int. Conf. Very Large Data Bases*, vol. 29. 2003, PP. 898–909.

[22] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, p. 18, 2009.

[23] T.-Y. Ma, T.-W. Hou, and S.-Y. Tseng, "Hierarchical key management of scalable video coding," in *Proc. 3rd Int. Conf. Intell. Inf. Hiding Multimedia Signal Process. (IIHMSP)*, vol. 1. 2007, pp. 399–402.

[24] E. Damiani, "Key management for multi-user encrypted databases," in *Proc. ACM Workshop Storage Secur. Survivability*, 2005, pp. 74–83.

[25] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: management of access control evolution on outsourced data," in *Proc. 33rd Int. Conf. Very Large Data Bases (VLDB)*, 2007, pp. 123–134.

[26] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 55–66.

[27] A. Yun, C. Shi, and Y. Kim, "On protecting integrity and confidentiality of cryptographic file system for outsourced storage," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 67–76.

[28] B. Lynn. *The Stanford Pairing Based Crypto Library*. [Online]. Available: https://crypto.stanford.edu/pbc/

[29] R. Huang, X. Gui, S. Yu, and W. Zhuang, "Research on privacy-preserving cloud storage framework supporting ciphertext retrieval," in *Proc. Int. Conf. Netw. Comput. Inf. Secur. (NCIS)*, vol. 1. 2011, pp. 93–97.

[30] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST, Gaithersburg, MD, USA, Tech. Rep. SP 800-145, 2011.

[31] Charm. [Online]. Available: http://www.charm-crypto.com/

[32] Z. Liu, Z. Cao, and D. S. Wong, "Efficient generation of linear secret sharing scheme matrices from threshold access trees," IACR Cryptology ePrint Archive, 2010.

[33] J. A. Akinyele, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

[34] V. Odelu, A. K. Das, Y. S. Rao, S. Kumari, M. K. Khan, and K.-K. R. Choo, "Pairing-based CP-ABE with constant-size ciphertexts and secret keys for cloud environment," *Comput. Standards Interfaces*, May 2016. [Online]. Available: http://dx.doi.org/10.1016/j.csi.2016.05.002
[35] V. Odelu and A. K. Das, "Design of a new CP-ABE with constant-size secret keys for lightweight devices using elliptic curve cryptography," *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4048–4059, 2016.
[36] S. Chatterjee and A. K. Das, "An effective ECC-based user access control scheme with attribute-based encryption for wireless sensor networks," *Secur. Commun. Netw.*, vol. 8, no. 9, pp. 1752–1771, 2015.

FAWAD KHAN received the B.S. degree in electrical engineering from UET Peshawar in 2010 and the M.S. degree in electrical engineering from CECOS University in 2014. He is currently pursuing the Ph.D. degree with the School of Cyber Engineering, Xidian University. He was with NUCES-FAST as a Lab Engineer from 2011 to 2015. His research interests include content centric networks, information security, and machine learning.

HUI LI (M'10) received the B.S. degree from Fudan University in 1990, and the M.S. and Ph.D. degrees from Xidian University in 1993 and 1998, respectively. In 2009, he was with Department of Electrical and Computer Engineering, University of Waterloo, as a Visiting Scholar. He is currently a Professor with the School of Cyber Engineering, Xidian University. His research interests include the areas of cryptography, security of cloud computing, wireless network security, and information theory. He served as the TPC Co-Chair of ISPEC 2009 and IAS 2009, and the General Co-Chair of E-Forensic 2010, ProvSec 2011, and ISC 2011.

LIANGXUAN ZHANG received the B.S. degree in mathematics from Xiangtan University in 2014. He is currently pursuing the master's degree with the School of Cyber Engineering, Xidian University. His current research interests include security and privacy issues in cloud computing and applied cryptography.