IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

Secure Cloud Storage via Deniable Attribute Based Encryption with Efficient Revocation

Aswathy S[1], Thahsin V P[1], Veena Gopakumar[1], Suja Rani[2]

Students, Information Technology, College of Engineering Perumon, Kollam, India[1]

Journal Guide, Information Technology, College Of Engineering Perumon, Kollam, India[2]

Abstract: Cloud computing, also known as 'on-demand computing', is a kind of internet based computing, where shared resources, data and information are provided to computers and other devices on demand. As cloud computing brings ease and cost-saving features, the security and privacy of data is simultaneously becoming very challenging. For providing privacy, many encryption schemes have been proposed. Most of them assume that it provides proper security and cannot be hacked by any unauthorized users, but in practice, some authorities may force cloud providers to reveal user secrets in some circumstances. Here in this paper we propose an encryption scheme which convinces the fake users by providing fake details to the unauthorized users and provide efficient revocation schemes. Thereby we can make our data more secure and private and can be protected from unauthorized users.

Keywords: ABE, CP-ABE, KP-ABE, LSSS.

I. INTRODUCTION

Nowadays, there is an emerging trend that increasingly more customers are beginning to use the cloud storage for online data storing and sharing[e]. Cloud storage becomes popular because of its ease of use and cost-saving features.Users can store their data and access their data anywhere at any time from the cloud.As cloud storing become famous its security is also a big challenge. For protecting the data in the cloud we use some encryption schemes to encrypt the data thereby we can protect the access of data from other users. The most common encryption scheme used for encryption is attribute-based encryption (ABE). There are numerous ABE schemes that have been proposed.

Most of these schemes assume that cloud providers provide proper management and can protect their personal information from other users. Once the users publish their private data to the cloud storage,they lose the direct control of their data and have to trust the cloud storage service provider. But in some circumstances the cloud providers compel to reveal user secrets by using some powers or to the Government in case of investigation etc. To protect their sensitive data, customers need to encrypt the data before sending to the cloud storage. Once the cloud storage providers are compromised, all encryption schemes lose their effectiveness.

In this work, we offer an encryption scheme and a revocation scheme to the cloud storage to protect user data from unauthorised users by creating fake user secrets. Using deniable encryption, unauthorized user can only obtainfake details from user's stored cipher text by convincing them that the data they get are real, thereby they are satisfied and cloud storage would not have to reveal any real secrets. In case they know that the given data is not real, they have no reason to reject the given data because they have no evidence to prove that the given data is not real since they know that their effects will be useless.

Here we prefer to use ciphertext policy-attribute based encryption (CP-ABE) for encryption. We enhance the Waters scheme from prime order bilinear group to Composite order bilinear groups. By the subgroup decision problem assumption, our scheme enables users to be able to provide fake secrets that seem legitimate to outside coercers. Here we also use efficient revocation along with the encryption schemes i.e., periodically change the secret key of the data owner and user and also re-encrypt the data stored in the cloud.

II. PREVIOUS WORK ON ABE

Sahai and Waters first introduced the concept of ABE in which data owners can embed how they want to share data in terms of encryption. That is, only those who match the owner's conditions can successfully decrypt stored data. We note here that ABE is encryption for privileges, not for users. This makes ABE a very useful tool for cloud storage services since data sharing is an important feature for such services. There are so many cloud storage users that it is impractical for data owners to encrypt their data by pairwise keys. Moreover, it is also impractical to encrypt data many times for many people. With ABE, data owners decide only which kind of users can access their encrypted data. Users who satisfy the conditions are able to decrypt the encrypted data.

There are two types of ABE, CP-ABE and Key-Policy ABE (KP-ABE). The difference between these two lies in policy checking. KP-ABE is an ABE in which the policy is embedded in the user secret key and the attribute set is embedded in the ciphertext. Conversely, CP-ABE embeds the policy into the ciphertext and the user secret has the attribute set. Goyal et al. proposed the first KPABE.

They constructed an expressive way to relate any monotonic formula as the policy for user secret keys. Bethencourt et al. proposed the first CP-ABE. This scheme used a tree access structure to express any monotonic formula over attributes as the policy in the ciphertext. The first fully expressive CP-ABE was proposed by Waters, which used Linear Secret Sharing Schemes (LSSS) to build a ciphertext policy. Lewko et al. enhanced the Waters scheme to a fully secure CP-ABE, though with some efficiency loss. Recently, Attrapadung et al. constructed a CP-ABE with a constant-size cipher text Tysowski et al. designed their CP-ABE scheme for resource-constrained users

III. PREVIOUS WORK ON DENIABLE ENCRYPTION

Like normal encryption schemes, deniable encryption can be divided into a deniable shared key scheme and a public key scheme. Considering the cloud storage scenario, we focus our efforts on the deniable public key encryption scheme.

There are some important deniable public key encryption schemes. Canetti et al. used translucent sets to construct deniable encryption schemes. A translucent set is a set containing a trapdoor subset. It is easy to randomly pick an element from the universal set or from the subset; however, without the trapdoor, it is difficult to determine if a given element belongs to the subset. Canetti et al. showed that any trapdoor permutation can be used to construct the translucent set. To build a deniable public key encryption scheme from a translucent set, the translucent set is the public key and the trapdoor is the private key. The translucent set is used to represent one encrypted bit. Elements in the subset are represented by 1 whereas other non-subset elements are represented by 0. The sender can encrypt 1 by sending an element in the subset, but can claim the element is chosen from the universal set (i.e., 0). The above is a basic sender-deniable scheme. Canetti et al. also proved that a sender-deniable scheme can be transformed to a receiver-deniable scheme or a deniable scheme with the help of intermediaries. There is research on how best to design a translucent set. Durmuth et al. designed the translucent set from the saleable encryption. ONeill et al. designed the bi-translucent set from a lattice, which can build a native bi-deniable scheme.

In addition to the bitranslucent set, there are other proposed approaches to building deniable encryption schemes. ONeill et al. proposed a new deniable method through a simulatable public key system. The simulatable public key system provides an oblivious key generation function and an oblivious ciphertext function. When sending an encrypted bit, the sender will send a set of encrypted data which may be normally encrypted or oblivious. Therefore, the sender can claim some sent messages are oblivious while actually they are not. The idea can be applied to the receiver side such that the scheme is a bi-deniable scheme. Gasti et al. proposed another deniable scheme in which one publicprivate key pair is set up for each user while there are actually two pairs. The sender can send a true message encrypted by one key with a fake message encrypted by the other key. The sender decides which key is released according to the coercer’s identity. Gasti et al. also applied this idea to cloud storage services.

Aside from the above deniable schemes, there is research investigating the limitations of the deniable schemes. Nielsen states that it is impossible to encrypt unbounded messages by one short key in non-committing schemes, including deniable schemes. Bendlin et al. shows that non interactive and fully receiver-deniable schemes cannot be achieved simultaneously. We construct our scheme under these limitations.

IV. PREVIOUS WORK ON REVOCATION SCHEME

In a fine-grained cryptographic access control system, the flexibility of the access policies is mainly restricted by the performance of the revocation solution. The simplest model for reducing there vocation’ sconsumption is named the lazy revocation scheme (Backes*et al.*, 2005). The main idea of this scheme is to postpone the data retrieval, re encryption, and re-publication operations until the data are updated. For instance, suppose that a DUis revoked to access the file *F*. DO will record this change in local storage instead of executing immediately. Lazy revocation is suitable for some unstrict situations where the files are allowed to remain encrypted withold keys and revocation operations take place occasionally(Blanchet and Chaudhuri, 2008; Kumbhare*et al.*, 2011). However, lazy revocation just evades the revocation problem, which cannot support the secure policies enforcing. In most actual cases, the new access policy needs to be enforced immediately after a revocation takes place. To support efficient but secure revocation, some researchers proposed key revocation (Yu *et al.*,2010b; Jahid*et al.*, 2011; Xu and Martin, 2012),proxy re-encryption (Liang *et al.*, 2009; Libert and Vergnaud, 2011), and over-encryption (di Vimercati*et al.*, 2007) schemes.

V. PRELIMINARIES

1. Prime Order Bilinear Groups

Let G and GT be two multiplicative cyclic groups of prime order p, with map function e : G × G → GT . Let g be a generator of GG. G is a bilinear map group if G and e have the following properties:

• Bilinearity: ∀u, v ∈ G and a, b ∈ Z, e(ua, vb) = e(u, v)ab.
• Non-degeneracy: e(g, g) 6= 1.
• Computability: the group action in G and map function e can be computed efficiently.

2. ASBE

ASBE’s capability of assigning multiple values to the same attribute enables it to solve the user revocation problem efficiently, which is difficult in CP-ABE. The revocation problem can be solved easily by assigning different expiration times. The above desirable feature and the recursive key structure is implemented by four algorithms they are , Setup, KeyGen, Encrypt, and Decrypt:

Setup algorithm (MK, PK) ← Setup (1k): is run by the trusted authority or the security administrator. The setup algorithm takes as input a security parameter k and outputs a master secret key MK and a master public key PK.

Key Generation algorithm (SK) ← Key Gen (MK, ω): is run by the trusted authority, and takes as input a set of attributes ω and MK. The algorithm outputs a user secret key SK associated with the attribute set ω**.**

Encryption algorithm (**CT**) **← Encrypt (*m*, PK, P)**: is run by the encryptor. For encryption the input of the algorithm is a message file *m*, a master public key PK and an access control policy P, the output of the algorithm is a ciphertext CT encrypted under the access control policy P.

Decryption algorithm (m) ← Decrypt (CT, SK): is run by the decryptor. The input of the algorithm is a ciphertext CT to be decrypted and a user secret key SK. The output of the algorithm is a message *m*, if the attribute set of the secret key satisfies the access policy P under which the message was encrypted, or an error message if the attribute set of the secret key does not satisfies the access policy P under which the message was encrypted. These algorithms are essentially similar to those of CP-ABE, except some extensions to support recursive key structure. The public key and the master key of ASBE are extended from CPABE to have components supporting recursive key structure. The master key is extended by adding a new secret exponent β*d* for depth. The generated private keys are also different in ASBE and CP-ABE. There are translating components that enable attributes translation between different key sets. The missing part of ASBE is the delegation algorithm, which is used in our proposed scheme to construct the hierarchical structure. We accept the same four algorithms of ASBE, and extend ASBE by proposing a new delegation algorithm.

Definition (Bilinear subgroup decision assumption): Let G be a 2-cancelling bilinear group generator such that for i = 1; 2 the output groups Gi and Hi are of prime order pi. We define the following distribution:

$$\mathbb{G} = (G, H, G_t, e, N := \mathrm{lcm}(p_1, p_2)) \overset{R}{\leftarrow} \mathcal{G}(\lambda),\ f_1, g_1 \overset{R}{\leftarrow} G_1,\ f_2, g_2 \overset{R}{\leftarrow} G_2,\ h_1 \overset{R}{\leftarrow} H_1,\ h_2 \overset{R}{\leftarrow} H_2,$$
$$Z \leftarrow (g_1, g_2, h_1, h_2),$$
$$T_0 \leftarrow e(f_2, h_1 h_2),\ T_1 \leftarrow e(f_1 f_2, h_1 h_2).$$

We define the advantage of an algorithm A in solving the bilinear subgroup decision assumption to be

$$\text{BSD-Adv}[\mathcal{A}, \mathcal{G}] = \left| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \right|.$$

We say that G satisfies the bilinear subgroup decision assumption if $\text{BSD-Adv}[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm A.

The reason the BSD assumption does not reduce to the ordinary subgroup decision assumption (with G1;G2 switched) is that in the latter the challenger is not given generators of both G1 and G2.

With these definitions, we can now state the security theorem for the generalized Boneh-Sahai- Waters scheme. The original proof applies in our more general context.

Theorem : Suppose that G satisfies the subgroup decision assumption on the right, the bilinear subgroup decision assumption, and the 3-party Di_e-Hellman assumptions on the left and right. Then the generalized Boneh-Sahai-Waters PLBE scheme is secure.

3. Chameleon Hash

A distinguishing feature of chameleon signature schemes is that they are non-transferable, i.e. a signature issued to a designated recipient cannot be validated by another party. While not universally verifiable, chameleon signatures provide non-repudiation: If presented with a false signature claim, the signer can prove that the signature is forged, while incapable of doing so for legitimate claims. Accordingly, the signer's refusal to invalidate a signature is considered equivalent to her affirmation that the signature is valid.

Unlike undeniable signatures, which also provide non-repudiation and non-transferability, chameleon signatures are non-interactive protocols. More precisely, the signer can generate the chameleon signature without interacting with the designated recipient, and the latter will be able to verify the signature without interacting with the former. Similarly, if presented with a forged signature, the signer can deny its validity by revealing certain values. These values will revoke the original signature and the forged one simultaneously, and the revocation can be universally verified. In other words, the forged-signature denial protocol is also non-interactive. There also exist non-interactive versions of undeniable signatures. Chameleon signatures are considerably less complex, at the sacrifice of not conferring the signer the ability to engage in non-transferable secondary proofs of signature (non-)validity. Chameleon signatures are based on the well established hash-and-sign paradigm, where a chameleon hash function is used to compute the cryptographic message digest. A chameleon hash function is a trapdoor one-way hash function: Without knowledge of the associated trapdoor, the chameleon hash function is resistant to the computation of pre-images and of collisions. However, with knowledge of the trapdoor, collisions are efficiently computable.

When a chameleon hash function is used within a hash-and-sign signature scheme, it permits the party with knowledge of the trapdoor to re-use the signature value to authenticate other messages of choice. In particular, if the hash function is part of the recipient's public key, then the signature is publicly verifiable by no one other than the intended recipient. On the other hand, if the recipient reuses the hash value to obtain a signature on a second message, the signer can prove knowledge of a hash collision, since the original signed message and the claimed signed message have the same hash value. Because computing hash collisions is infeasible for the signer, possession of such a collision is seen as proof of forgery by the signature recipient.

VI. DEFINITION

A. Deniable CP-ABE Scheme

Deniable encryption schemes may have different properties and we provide an introduction to many of these properties below.

- ad hoc deniability vs. plan-ahead deniability: The former can generate a fake message (from the entire message space) when coerced, whereas the latter requires a predetermined fakemessage for encryption. Undoubtedly, all bitwise encryption schemes are ad hoc.
- sender-, receiver-, and bi-deniability: The prefix here in each case implies the role that can fool the coercer with convincing fake evidence. In sender-deniable encryption schemes and receiver-deniable schemes, it is assumed that the other entity cannot be coerced. Bi-deniability means both sender and receiver can generate fake evidence to pass third-party coercion.
- full deniability vs. multi-distributional deniability: A fully deniable encryption scheme is one in which there is only one set of algorithms, i.e., a keygeneration algorithm, an encryption algorithm and so on. Senders, receivers and coercers know this set of algorithms and a sender and a receiver can fool a coercer under this condition. As for multi-distributional deniable encryption schemes, there are two sets of algorithms, one while the other is a deniable set. The outputs of algorithms in these two sets are computationally indistinguishable. The normal set of algorithms cannot be used to fool coercers, whereas the deniable set can be used. A sender and a receiver can use the deniable algorithm set, but claim that they use the normal algorithm set to fool coercers.
- interactive encryption vs. non-interactive encryption: The difference between these two types of encryption is that the latter scheme does not need interaction between sender and receiver.

According to the above definitions, the ideal deniable encryption scheme is ad hoc, full, bi-deniability and non-interactive deniability; however, there is research focused on determining the limitations of the deniable schemes. Nielsen stated that it is impossible to encrypt unbounded messages by one short key in non-committing schemes, including deniable schemes. Since we want our scheme to be block wise deniable with a consistent encryption environment, we design our scheme to be a plan-ahead deniable encryption scheme. Bendlin et al. showed that non-interactive and fully receiver deniable properties cannot be achieved simultaneously. We prefer our scheme to have the non-interactive property for ease of use. Therefore, our scheme is multi-distributional. In summary, our deniable scheme is planahead, bi-deniable, and multi-distributional. Below, we provide the definition of this kind of deniable CP-ASBE scheme.

Definition (Deniable CP-ASBE): Our plan-ahead, bideniable, and multi-distributional CP-ASBE scheme is composed of the following algorithms:

- **Setup**(1_) → (PP,MK): This algorithm takes security parameter _ as input and returns public parameter PP and system master key MK.
- **KeyGen**(MK, S) → SK: Given set of attributes S and MK, this algorithm outputs private key SK.
- **Enc**(PP,M,T) → CT: This encryption algorithm takes as input public parameter PP, message M, and access tree T = (M, _) over the universe of attributes. This algorithm encrypts M and outputs a ciphertext CT, which can be decrypted by those who possess an attribute set that satisfies access tree T. Note that T is contained in CT.
- **Dec**(CT,SK) → {M,⊥}: This decryption algorithm takes as input public parameter PP, private key SK with its attribute set S, and ciphertext CT with its access tree T. If S satisfies T, then this algorithm returns M; otherwise, this algorithm returns ⊥.
- **OpenEnc**(PP,CT,M) → PE: This algorithm is for the sender to release encryption proof PE for (M,CT).
- **OpenDec**(PP, SK,CT,M) → PD: This algorithm is for the receiver to release decryption proof PD for (M,CT).
- **Verify**(PP,CT,M, PE, PD) → {T, F}: This algorithm is used to verify the correctness of PE and PD.
- **DenSetup**(1_) → (PP,MK, PK): This algorithm takes security parameter _ as input and returns public parameters PP, system master key MK, and system public key PK. PK is known by all system users and is kept secret to outsiders.
- **DenKeyGen**(MK, S) → (SK, FK): Given set of attributes S and MK, this algorithm outputs private key SK as well as FK for the user, where FK will be used for generating fake proof later.
- **DenEnc**(PP, PK,M,M′,T) → CT′: Aside from the inputs of the normal encryption algorithm, this deniable encryption algorithm needs public key PK and fake message M′. The output ciphertext must be indistinguishable from the output of **Enc**.
- **DenOpenEnc**(PP,CT′,M′) → P′ E : This algorithm isfor the sender to release encryption proof P′E for fake message M′. The output must be indistinguishable from the result of **OpenEnc**and must pass the **Verify** algorithm.
- **DenOpenDec**(PP, SK, FK,CT′,M′) → P′ D: This algorithmis for the receiver to release decryptionproof P′ D for fake message M′. The output must beindistinguishable from the result of **OpenDec**andmust pass the **Verify** algorithm.

VII. DENIABLE CP-ABE CONSTRUCTION

Let $\mathbb{G}_0$ be a bilinear group of prime order p and let g be a generator of $\mathbb{G}_0$. Let

$e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ denote a bilinear map. Let $H : \{0,1\}^* \rightarrow \mathbb{G}_0$ be a hash function that maps any arbitrary string to a random group element. We will use this function to map attributes described as arbitrary strings to group elements.

Setup(d = 2). The setup algorithm chooses random exponents $\alpha, \beta_i \in \mathbb{Z}_p \forall i \in \{1, 2\}.$
The algorithm sets the public key and master key as:
PK=
$(\mathbb{G}, g, h_1| = g^{\beta_1}, f_1 = g^{\frac{1}{\beta_1}}, h_2 = g^{\beta_2}, f_2 = g^{\frac{1}{\beta_2}}, e(g,g)^{\alpha})$

MK= $(\beta_1, \beta_2, g^{\alpha})$
Note that to support key structures of depth d, i will range from 1 to d.

KeyGen(MK, A, u). Here u is the identity of a user and A = {A0,A1, . . . ,Am} is a key structure. A0 is the set of individual attributes in the outer set (i.e. set 0) and A1 to Am are sets of attributes at depth 2 that the user has. Let Ai = {ai,1, . . . , ai,ni}. That is, ai,j denotes the j-th attribute appearing in set Ai, and ni denotes the number of attributes in the set Ai. (Note that for different values of (i, j), ai,j can be the same attribute.) The key generation algorithm chooses a unique random number, r{u} 2 Zp, for user u. It then chooses a set of m unique random numbers, $r_i^{\{u\}} \in \mathbb{Z}_p$, one for each set $A_i \in \mathbb{A}, 1 \leq i \leq m$. For set A0, r{u} 0 is set to be the same as r{u}. It also chooses a set of unique random numbers, $r_{i,j}^{\{u\}} \in \mathbb{Z}_p,$ one for each $(i, j), 0 \leq i \leq m, 1 \leq j \leq n_i.$

.

The issued key is:

$$\begin{aligned} \mathrm{SK}_u = \Big(\mathbb{A}, D = g^{\frac{(\alpha + r^{\{u\}})}{\beta_1}}, \\ D_{i,j} = g^{r_i^{\{u\}}} \cdot H(a_{i,j})^{r_{i,j}^{\{u\}}}, D'_{i,j} = g^{r_{i,j}^{\{u\}}} \quad \text{for } 0 \leq i \leq m, 1 \leq j \leq n_i, \\ E_i = g^{\frac{(r^{\{u\}} + r_i^{\{u\}})}{\beta_2}} \quad \text{for } 1 \leq i \leq m \Big) \end{aligned}$$

Note that the operations on the exponents in the above equations are modulo the order of the group, which is prime. Hence division in the exponent is well-defined. We omit the mod for convenience. Elements Ei enable translation from r{u}i (i.e., set Ai at depth 2) to r{u} (i.e., the outer or parent set A0 at depth 1) at the translating nodes. Elements Ei and Ei0 can be combined as Ei/Ei0 to enable translation from r{u} i0 (i.e., set Ai0) to r{u} i (i.e., the set Ai) at the translating nodes. Similarly, for a key structure of depth d, there will elements that enable translation from a set at depth d to its parent set at depth
d− 1 and they will use _d and random numbers corresponding to the appropriate sets.

Encrypt(PK, M, T). M is the message, T is an access tree. The algorithm associates a polynomial $q_{\mathcal{T}}$ with each node T (including the leaves) in the tree T . These polynomials are chosen in the following way in a top-down manner, starting from the root node R. For each internal node T in the tree, the degree d_ of the polynomial $q_{\mathcal{T}}$ is set to be one less than the threshold value k_ of that node, that is, $d_{\mathcal{T}} = k_{\mathcal{T}} - 1.$. For leaf nodes the degree is set to be 0. For the root node R the algorithm picks a random $s \in Z_p$ and sets qR(0) = s. Then, it chooses dR other points randomly to define the polynomial qR completely. For any other node T , it sets $q_{\mathcal{T}}(0) = q_{parent(\mathcal{T})}(index(\mathcal{T}))$ and chooses d T other points randomly to completely define $q_{\mathcal{T}}$. Here parent(T) denotes the parent node of T . Let Y denote the set of leaf nodes in T . Let X denote the set of translating nodes in the access tree T . Then the ciphertext CT returned is as follows:

$$\begin{aligned} \mathrm{CT} = (\mathcal{T}, \tilde{C} = M \cdot e(g,g)^{\alpha \cdot s}, C = h_1^s, \bar{C} = h_2^s, \forall y \in \mathbb{Y} : C_y = g^{q_y(0)}, \\ C'_y = H(att(y))^{q_y(0)}, \forall x \in \mathbb{X} : \hat{C}_x = h_2^{q_x(0)}) \end{aligned}$$

Translating values $\hat{C}'_x$ together with $E_i{}'$ in user keys allow translation between sets ata translating node x as will be described in the Decrypt function. Note that the element
$\bar{C}$ is the same as $\hat{C}_r$ where r denotes the root node. A variant of the scheme would
be where ¯ C is not included in the ciphertext but is only released at the discretion of the encrypting user as $\hat{C}_r$. This would restrict decrypting users to only use individual attributes in the outer set except when explicitly allowed by the encrypting user by designating translating nodes.

Decrypt(CT, SKu). Here we describe the most straightforward decryption algorithm without regard to efficiency. The decryption algorithm is a recursive algorithm similar to the tree satisfaction algorithm described in Section 4.

The decryption algorithm first runs the tree satisfaction algorithm on the access tree with the key structure i.e., T (A), and stores the results of each of the recursive calls in the access tree T . That is, each node t in the tree is associated with a set St of labels that was returned by Tt(A). If A does not satisfy the tree T then the decryption algorithm returns ?. Otherwise the decryption algorithm picks one of the labels, i, from the set returned by T (A) and calls a recursive function.

DecryptNode(CT, SKu, t, i) on the root node of the tree. Here CT is the ciphertext $\mathrm{CT} = (\mathcal{T}, \tilde{C}, C, \forall y \in Y : C_y, C'_y, \forall x \in X : \hat{C}_x)$, SKu is a private key, which is associated with a key structure denoted by A, t is a node from T , and i is a label denoting a set of A. Note that the ciphertext CT now contains tree information that is augmented by the results from T (A). DecryptNode(CT, SKu, t, i) is defined as follows.

If $t \in \mathbb{Y}$, i.e., node t is a leaf node, then **DecryptNode**(CT, SKu, t, i) is defined as follows. If $att(t) \notin A_i$ where $A_i \in \mathbb{A}$ then DecryptNode(CT, SKu, t, i) $= \perp$. If $att(t) = a_{i,j} \in A_i$ where $A_i \in \mathbb{A}$ then:

$$DecryptNode(\mathrm{CT}, \mathrm{SK}_u, t, i) = \frac{e(D_{i,j}, C_t)}{e(D'_{i,j}, C'_t)} = \frac{e(g^{r_i^{\{u\}}} \cdot H(a_{i,j})^{r_{i,j}^{\{u\}}}, g^{q_t(0)})}{e(g^{r_{i,j}^{\{u\}}}, H(a_{i,j})^{q_t(0)})} = e(g,g)^{r_i^{\{u\}} \cdot q_t(0)}$$

Note that set from which the satisfying attribute ai,j was picked is implicit in the result $e(g,g)^{r_i^{\{u\}} \cdot q_t(0)}$ (i.e., indicated by r{u}i). When $t \notin \mathbb{Y}$, i.e., node t is a non-leaf node, thenDecryptNode(CT, SKu, t, i) proceeds as follows:

1. Compute Bt which is an arbitrary kt sized set of child nodes z such that z 2 Bt only if either (1) label $i \in S_z$ or (2) label $i' \in S_z$ for some $i' \neq i$ and z is a translating node. If no such set exists then return $\perp$.

2. For each node z $\in$Bt such that label i $\in$Sz call DecryptNode(CT, SKu, t, i)
and store output in Fz.

3. For each node z $\in$Bt such that i0 $\in$Sz and callDecryptNode(CT, SKu, t, i0)
store output in F0 . If $i \neq 0$ then translate F0 to Fz as follows:

$$F_z = e(\hat{C}_z, E_i/E_{i'}) \cdot F'_z$$
$$= e(g^{\beta_2 \cdot q_z(0)}, g^{\frac{r_i^{\{u\}} - r_{i'}^{\{u\}}}{\beta_2}}) \cdot e(g,g)^{r^{\{u\}}{}_{i'} \cdot q_z(0)} = e(g,g)^{r^{\{u\}}i \cdot q_z(0)}$$

Otherwise, translate F'_z to Fz as follows:

$$F_z = \frac{e(\hat{C}_z, E_{i'})}{F'_z} = \frac{e(g^{\beta_2 \cdot q_z(0)}, g^{\frac{r^{\{u\}} + r_{i'}^{\{u\}}}{\beta_2}})}{e(g,g)^{r_{i'}^{\{u\}} \cdot q_z(0)}} = e(g,g)^{r^{\{u\}} \cdot q_z(0)}$$

4. Compute Ft using polynomial interpolation in the exponent as follows:

$$F_t = \prod_{z \in B_t} F_z^{\Delta_{k,B'_z}(0)}, \qquad \text{where } k = index(z), B'_z = \{index(z) : z \in B_t\}$$

$$\text{and Lagrange coefficient } \Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

$$= \begin{cases} e(g,g)^{r_i^{\{u\}} \cdot q_t(0)} & \text{when } i \neq 0 \\ e(g,g)^{r^{\{u\}} \cdot q_t(0)} & \text{when } i = 0 \end{cases}$$

The output of DecryptNode(CT, SKu, r, i) function on the root node r is stored in Fr. If i = 0 we have Fr = e(g, g)r{u}·qr(0) = e(g, g)r{u}·s otherwise we have Fr = e(g, g)r{u} i ·s. If i 6= 0 then we compute F as follows:

$$F = \frac{e(\hat{C}_r, E_i)}{F_r} = \frac{e(g^{\beta_2 \cdot q_r(0)}, g^{\frac{r^{\{u\}} + r_i^{\{u\}}}{\beta_2}})}{e(g,g)^{r_i^{\{u\}} \cdot q_r(0)}} = e(g,g)^{r^{\{u\}} \cdot q_r(0)} = e(g,g)^{r^{\{u\}} \cdot s}$$

Otherwise F = Fr. The decryption algorithm then computes following:

$$\frac{\tilde{C} \cdot F}{e(C,D)} = \frac{M \cdot e(g,g)^{\alpha \cdot s} \cdot e(g,g)^{r^{\{u\}} \cdot s}}{e(g^{s \cdot \beta_1}, g^{\frac{(r^{\{u\}} + \alpha)}{\beta_1}})} = M$$

Note how two elements Ei and Ei0 together with a translating value $\hat{C}_t$ at a node t were used to translate between sets i and i0 at node t in step 3. Similarly, note how a single element Ei together with a translating value was used to translate between set i and the outer set. We note that if $\beta_1 = \beta_2$ then the scheme would become insecure as colluding users could transitively translate from inner set i to outer set and then from one key to the other by using the D elements from their keys. Thus we need a unique β for every level that we need to support. When using key structures of depth d, translating values, $\hat{C}s$, that help translate between sets at depth d or between a set
at depth d and its parent at depth d − 1 will use β_d. And to allow translations across multiple levels at a given node, multiple translating values using different βs will need to be released at that node.

- **OpenEnc**(PP,C,M) → PE: This algorithm returns two coins b0, b1 as proof PE.
- **OpenDec**(PP, SK,C,M) → PD: This algorithm directly returns SK as proof PD since this is the most persuasive proof.
- **Verify**(PP,C,M, PE, PD) → {T, F}: To verify PE and PD, this algorithm first runs Dec(PP, PD,C) and checks if the output is equal to declared input M. Then, this algorithm checks PE with correct coins b0, b1 derived in the decryption process. If bothrequirements are satisfied, this algorithm returns T ;otherwise, it returns F.
- **DenSetup**(1^λ) → (PP,MSK, PK): This algorithm runs Setup(1^λ) and obtains PP. System public key PK is $\{g_2g_3, (g_2g_3)^\alpha, e(g_3,g_3)^\alpha, e(g_2g_3, g_2g_3)^\alpha\}$ and system secret key MSK is $\{(g_1g_3)^\alpha, g_1g_2g_3, (g_1g_2g_3)^\alpha\}$.
- **DenKeyGen**(MSK, S) → (SK, FK): This algorithm runs KeyGenand obtains SK for S. Next, this algorithm picks t′ ∈ZN and generates FK as follows:

$$\begin{aligned} FK &= \{(g_1g_2g_3)^{\alpha+at'}, (g_1g_2g_3)^{t'}, \{H_1(x)^{t'}\}_{\forall x \in S}\} \\ &= \{K', L', \{K'_x\}_{\forall x \in S}\}. \end{aligned}$$

- **DenEnc**(PP, PK,M,M′,A = (M,ρ)) → C′: This algorithm prepares $\lambda_i, \forall i \in \{1,\ldots,l\}$ just as the Enc algorithm does. This algorithm sets up chameleon hash function CH(• , •). The chameleon hash function is determined during encryption. Note that without the trapdoor, a chameleon hash is just a one-way hash function. That is, a sender can claim this is just a normal hash function without any trapdoor. Output deniable ciphertext C′ will be:

$$C' = \{A'_0, A'_1, B', (C'_1, D'_1), \ldots, (C'_l, D'_l), CH, t_0, t_1, V\},$$

where,

$$A'_{b_0} = \mathcal{M} \cdot e(g_3, g_3)^{\alpha s}, A'_{1-b_0} = \mathcal{M}' \cdot e(g_2 g_3, g_2 g_3)^{\alpha s},$$
$$B' = (g_2 g_3)^s,$$
$$C'_i = (g_2 g_3)^{a\lambda_i} H_1(\rho(i))^{-r_i}, D'_i = (g_1 g_3)^{r_i}, i = 1, \ldots, l,$$
$$V = CH(\mathcal{M}, t_{b_1}) = CH(\mathcal{M}', t_{1-b_1}).$$

Based on the property of the chameleon hash, the sender can easily find tb1 and t1−b1 satisfying the above requirements.

- **DenOpenEnc**(PP,C′,M′) → P′ E : When the sendertries to fool the coercer with the pre-determined fake message, this algorithm returns two coins 1−b1, 1−b2 as its proof P′ E .
- **DenOpenDec**(PP, SK, FK,C′,M′) → P′ D: This algorithmdirectly returns FK as proof P′ D

User Revocation: Whenever there is a user to be invalidated, the system must make sure the invalidated user cannot access the associated data files any more. One way to solve this problem is to re-encrypt all the associated data files used to be accessed by the revoked user, but we must also confirm that the other users who still have access privileges to these data files can access them correctly.We add an attribute expiration-time X to a user's key, which indicates the time until which the key is considered to be valid. Then the policy linked with data files can include a check on the attribute expiration-time Y as a numerical comparison. The update of user's key and re-encryption of data files can be done as follows:

Key Update. Suppose that there is a user u, who is administrated by the domain authority DAi. DAipreserves some state information about u's key and adds a new value of expiration-time to u's existing key when it wants to update u's key. Then DAicomputes the secret key components corresponding to the expiration-time attribute and sends them to u. Transmission of the secret key components to the user can be accomplished with an out-of-band channel between DAiand the user. While DAiis required to maintain some state information about user's key, DAiavoids the need to create and distribute the entire keys on a frequent basis. This reduces the capacity on DAiand saves considerable computing resources.

Data Re-encryption. When the data owner wants to re-encrypt a data file, he changes the value of expiration-time the attribute in the key policy and computes the new ciphertext components cyand c'y, where is the leaf node on the access tree corresponding to the expiration-time attribute. Then the data owner sends these new ciphertext components to the cloud and the cloud service

Definition 6.2 (Bilinear subgroup decision assumption). Let G be a 2-cancelling bilinear
group generator such that for i = 1; 2 the output groups Gi and Hi are of prime order pi. We define the following distribution:

$$\mathbb{G} = (G, H, G_t, e, N := \mathrm{lcm}(p_1, p_2)) \overset{R}{\leftarrow} \mathcal{G}(\lambda),\ f_1, g_1 \overset{R}{\leftarrow} G_1,\ f_2, g_2 \overset{R}{\leftarrow} G_2,\ h_1 \overset{R}{\leftarrow} H_1,\ h_2 \overset{R}{\leftarrow} H_2,$$
$$Z \leftarrow (g_1, g_2, h_1, h_2),$$
$$T_0 \leftarrow e(f_2, h_1 h_2),\ T_1 \leftarrow e(f_1 f_2, h_1 h_2).$$

We define the advantage of an algorithm A in solving the bilinear subgroup decision assumption to be

$$\text{BSD-Adv}[\mathcal{A}, \mathcal{G}] = \left| \Pr[\mathcal{A}(\mathbb{G}, Z, T_0) = 1] - \Pr[\mathcal{A}(\mathbb{G}, Z, T_1) = 1] \right|.$$

We say that G satisfies the bilinear subgroup decision assumption if $\text{BSD-Adv}[\mathcal{A}, \mathcal{G}](\lambda)$ is a negligible function of λ for any polynomial-time algorithm A.

The reason the BSD assumption does not reduce to the ordinary subgroup decision assumption (with G1;G2 switched) is that in the latter the challenger is not given generators of both G1 and G2.

With these definitions, we can now state the security theorem for the generalized Boneh-Sahai- Waters scheme. The original proof applies in our more general context.

Theorem 6.3. Suppose that G satisfies the subgroup decision assumption on the right, the bilinear subgroup decision assumption, and the 3-party Di_e-Hellman assumptions on the left and right. Then the generalized Boneh-Sahai-Waters PLBE scheme is secure.

Definition 7 (Deniable CP-ASBE): Our plan-ahead, bideniable, and multi-distributional CP-ASBE scheme is composed of the following algorithms:

- **Setup**(1_) → (PP,MK): This algorithm takes security parameter _ as input and returns public parameter PP and system master key MK.
- **KeyGen**(MK, S) → SK: Given set of attributes S and MK, this algorithm outputs private key SK.
- **Enc**(PP,M,T) → CT: This encryption algorithm takes as input public parameter PP, message M, and access tree T = (M, _) over the universe of attributes. This algorithm encrypts M and outputs a ciphertext CT, which can be decrypted by those who possess an attribute set that satisfies access tree T. Note that T is contained in CT.
- **Dec**(CT,SK) → {M,⊥}: This decryption algorithm takes as input public parameter PP, private key SK with its attribute set S, and ciphertext CT with its access tree T. If S satisfies T, then this algorithm returns M; otherwise, this algorithm returns ⊥.

- **OpenEnc**(PP,CT,M) → PE: This algorithm is for the sender to release encryption proof PE for (M,CT).
- **OpenDec**(PP, SK,CT,M) → PD: This algorithm is for the receiver to release decryption proof PD for (M,CT).
- **Verify**(PP,CT,M, PE, PD) → {T, F}: This algorithm is used to verify the correctness of PE and PD.
- **DenSetup**(1_) → (PP,MK, PK): This algorithm takes security parameter _ as input and returns public parameters PP, system master key MK, and system public key PK. PK is known by all system users and is kept secret to outsiders.
- **DenKeyGen**(MK, S) → (SK, FK): Given set of attributes S and MK, this algorithm outputs private key SK as well as FK for the user, where FK will be used for generating fake proof later.
- **DenEnc**(PP, PK,M,M′,T) → CT′: Aside from the inputs of the normal encryption algorithm, this deniable encryption algorithm needs public key PK and fake message M′. The output ciphertext must be indistinguishable from the output of **Enc**.
- **DenOpenEnc**(PP,CT′,M′) → P′ E : This algorithm isfor the sender to release encryption proof P′ E for fake message M′. The output must be indistinguishable from the result of **OpenEnc**and must pass the **Verify** algorithm.
- **DenOpenDec**(PP, SK, FK,CT′,M′) → P′ D: This algorithmis for the receiver to release decryptionproof P′ D for fake message M′. The output must beindistinguishable from the result of **OpenDec**andmust pass the **Verify** algorithm.

VIII.CONCLUSION

ASBE's capability of assigning multiple values to the same attribute enables it to solve the user revocation problem efficiently, which is difficult in CP-ABE. The revocation problem can be solved easily by assigning different expiration times. The above desirable feature and the recursive key structure is implemented by four algorithms they are , Setup, KeyGen, Encrypt, and Decrypt:

Setup algorithm (MK, PK) ← Setup (1k): is run by the trusted authority or the security administrator. The setup algorithm takes as input a security parameter k and outputs a master secret key MK and a master public key PK.

Key Generation algorithm (SK) ← Key Gen (MK, ω): is run by the trusted authority, and takes as input a set of attributes ω and MK. The algorithm outputs a user secret key SK associated with the attribute set ω**.**

Encryption algorithm (**CT**) **← Encrypt (m, PK, P)**: is run by the encryptor. For encryption the input of the algorithm is a message file m, a master public key PK and an access control policy P, the output of the algorithm is a ciphertext CT encrypted under the access control policy P.

Decryption algorithm (m) ← Decrypt (CT, SK): is run by the decryptor. The input of the algorithm is a ciphertext CT to be decrypted and a user secret key SK. The output of the algorithm is a message m, if the attribute set of the secret key satisfies the access policy P under which the message was encrypted, or an error message if the attribute set of the secret key does not satisfies the access policy P under which the message was encrypted. These algorithms are essentially similar to those of CP-ABE, except some extensions to support recursive key structure. The public key and the master key of ASBE are extended from CPABE to have components supporting recursive key structure. The master key is extended by adding a new secret exponent βd for depth. The generated private keys are also different in ASBE and CP-ABE. There are translating components that enable attributes translation between different key sets. The missing part of ASBE is the delegation algorithm, which is used in our proposed scheme to construct the hierarchical structure. We accept the same four algorithms of ASBE, and extend ASBE by proposing a new delegation algorithm.

REFERENCES

[1]. A. Sahai and B. Waters, "Fuzzy identity-based encryption," in Eurocrypt, 2005, pp. 457–473.

[2]. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in ACM Conference on Computer and Communications Security, 2006, pp. 89–98.

[3]. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in IEEE Symposium on Security andPrivacy, 2007, pp. 321–334.

[4]. B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in Public KeyCryptography, 2011, pp. 53–70.

[5]. A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in Crypto, 2012, pp. 199–217.