This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

Audit-Free Cloud Storage via Deniable Attribute-based Encryption

Po-Wen Chi and Chin-Laung Lei, *Member, IEEE*

Abstract—Cloud storage services have become increasingly popular. Because of the importance of privacy, many cloud storage encryption schemes have been proposed to protect data from those who do not have access. All such schemes assumed that cloud storage providers are safe and cannot be hacked; however, in practice, some authorities (i.e., coercers) may force cloud storage providers to reveal user secrets or confidential data on the cloud, thus altogether circumventing storage encryption schemes. In this paper, we present our design for a new cloud storage encryption scheme that enables cloud storage providers to create convincing fake user secrets to protect user privacy. Since coercers cannot tell if obtained secrets are true or not, the cloud storage providers ensure that user privacy is still securely protected.

Index Terms—Deniable Encryption, Composite Order Bilinear Group, Attribute-Based Encryption, Cloud Storage.

1 Introduction

Cloud storage services have rapidly become increasingly popular. Users can store their data on the cloud and access their data anywhere at any time. Because of user privacy, the data stored on the cloud is typically encrypted and protected from access by other users. Considering the collaborative property of the cloud data, attribute-based encryption (ABE) is regarded as one of the most suitable encryption schemes for cloud storage. There are numerous ABE schemes that have been proposed, including [1], [2], [3], [4], [5], [6], [7].

Most of the proposed schemes assume cloud storage service providers or trusted third parties handling key management are trusted and cannot be hacked; however, in practice, some entities may intercept communications between users and cloud storage providers and then compel storage providers to release user secrets by using government power or other means. In this case, encrypted data are assumed to be known and storage providers are requested to release user secrets. As an example, in 2010, without notifying its users, Google released user documents to the FBI after receiving a search warrant [8]. In 2013, Edward Snowden disclosed the existence of global surveillance programs that collect such cloud data as emails, texts, and voice messages from some technology companies [9], [10]. Once cloud storage providers are compromised, all encryption schemes lose their effectiveness. Though we hope cloud storage providers can fight against such entities to maintain user privacy through legal avenues, it is seemingly more and more difficult. As one example, Lavabit was an email service company that protected all user emails from outside coercion; unfortunately, it failed and decided to shut down its email service [11].

Since it is difficult to fight against outside coercion, we aimed to build an encryption scheme that could help cloud storage providers avoid this predicament. In our approach, we offer cloud storage providers means to create fake user secrets. Given such fake user secrets, outside coercers can only obtained forged data from a user's stored ciphertext. Once coercers think the received secrets are real, they will be satisfied and more importantly cloud storage providers will not have revealed any real secrets. Therefore, user privacy is still protected.

This concept comes from a special kind of encryption scheme called **deniable encryption**, first proposed in [12]. Deniable encryption involves senders and receivers creating convincing fake evidence of forged data in ciphertexts such that outside coercers are satisfied. Note that deniability comes from the fact that coercers cannot prove the proposed evidence is wrong and therefore have no reason to reject the given evidence. This approach tries to altogether block coercion efforts since coercers know that their efforts will be useless. We make use of this idea such that cloud storage providers can provide audit-free storage services. In the cloud storage scenario, data owners who store their data on the cloud are just like senders in the deniable encryption scheme. Those who can access the encrypted data play the role of receiver in the deniable encryption scheme, including the cloud storage providers themselves, who have system-wide secrets and must be able to decrypt all encrypted data[1].

In this work, we describe a deniable ABE scheme for

• *The authors are with the Distributed Computing and Network Security (DCNS) Laboratory, Department of Electrical Engineering, National Taiwan University, Taiwan.*
E-mail: {d99921015,cllei}@ntu.edu.tw

1. Some papers divide this role into service providers and trusted key managers. More specifically, one is for cloud service operation, while the other is for key management and is assumed to be trusted. In this work we use cloud storage providers for both functions for simplicity. Further, this is also a common case in practice. Note that it is not difficult to apply our scheme to an architecture that has these two different roles defined.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

cloud storage services. We make use of ABE characteristics for securing stored data with a fine-grained access control mechanism and deniable encryption to prevent outside auditing. Our scheme is based on Waters ciphertext policy-attribute based encryption (CP-ABE) scheme [4]. We enhance the Waters scheme from prime order bilinear groups 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.

1.1 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 [1]. 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 KP-ABE in [2]. 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 in [3]. 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 in [4], 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, in [13]. Recently, Attrapadung et al. constructed a CP-ABE with a constant-size ciphertext in [14] and Tysowski et al. designed their CP-ABE scheme for resource-constrained users in [7].

1.2 Previous Work on Deniable Encryption

The concept of deniable encryption was first proposed in [12]. 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[2]. Canetti et al. used translucent sets to construct deniable encryption schemes in [12]. 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 bi-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 samplable encryption in [15]. ONeill et al. designed the bi-translucent set from a lattice in [16], 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 [16]. 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. In [17], Gasti et al. proposed another deniable scheme in which one public-private 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. There are still other deniable encryption schemes, including [18] and [19].

Aside from the above deniable schemes, there is research investigating the limitations of the deniable schemes. In [20]. Nielsen states that it is impossible to encrypt unbounded messages by one short key in non-committing schemes, including deniable schemes. In [21], Bendlin et al. shows that noninteractive and fully receiver-deniable schemes cannot be achieved simultaneously. We construct our scheme under these limitations.

2. For simplicity, in this paper deniable encryption means deniable public key encryption.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

1.3 Our Contributions

In this work, we construct a deniable CP-ABE scheme that can make cloud storage services secure and audit-free. In this scenario, cloud storage service providers are just regarded as receivers in other deniable schemes.

Unlike most previous deniable encryption schemes, we do not use translucent sets or simulatable public key systems to implement deniability. Instead, we adopt the idea proposed in [17] with some improvements. We construct our deniable encryption scheme through a multidimensional space. All data are encrypted into the multidimensional space. Only with the correct composition of dimensions is the original data obtainable. With false composition, ciphertexts will be decrypted to predetermined fake data. The information defining the dimensions is kept secret. We make use of composite order bilinear groups to construct the multidimensional space. We also use chameleon hash functions to make both true and fake messages convincing.

Our deniable ABE has the advantages described below over previous deniable encryption schemes.

- **Blockwise Deniable ABE**. Most deniable public key schemes (e.g., [12], [15], [16]) are bitwise, which means these schemes can only process one bit a time; therefore, bitwise deniable encryption schemes are inefficient for real use, especially in the cloud storage service case. To solve this problem, O'Neil et al. designed a hybrid encryption scheme that simultaneously uses symmetric and asymmetric encryption. They use a deniably encrypted plan-ahead symmetric data encryption key, while real data are encrypted by a symmetric key encryption mechanism. This reduces the repeating number from the block size to the key size. Though bitwise deniable encryption is more flexible than blockwise deniable encryption in "cooking" fake data, when considering cloud storage services, blockwise encryption is much more efficient in use.
  Unlike those techniques used in previous deniable encryption schemes, we build two encryption environments at the same time, much like the idea proposed in [17]. We build our scheme with multiple dimensions while claiming there is only one dimension. This approach removes obvious redundant parts in [17]. We apply this idea to an existing ABE scheme by replacing prime order groups with composite order groups. Since the base ABE scheme can encrypt one block each time, our deniable CP-ABE is certainly a blockwise deniable encryption scheme. Though the bilinear operation for the composite order group is slower than the prime order group, there are some techniques that can convert an encryption scheme from composite order groups to prime order groups for better computational performance, such as those described in [22] and [23]. We use composite order groups to describe our idea in Section 4 and transform it to prime order groups in Section 5.
- **Consistent Environment**. Most of the previous deniable encryption schemes are inter-encryption-independent. That is, the encryption parameters should be totally different for each encryption operation. If two deniable encryptions are performed in the same environment, the latter encryption will lose deniability after the first encryption is coerced, because each coercion will reduce flexibility. For example, once coercers get private keys, which are the most common receiver proofs, these keys should be convincing not only under some particular files, but also under all related stored data. Otherwise, the coercers will know that these keys are fake; however, all proposed schemes only provide convincing proofs for particular transmissions. In the secure cloud storage service, this is not practical. It is impossible for a cloud storage service provider to prepare a unique encryption environment for each file, much less to maintain the access control mechanism at the same time.
  In this work, we build a consistent environment for our deniable encryption scheme. By consistent environment, we means that one encryption environment can be used for multiple encryption times without system updates. The opened receiver proof should look convincing for all ciphertexts under this environment[3], regardless of whether a ciphertext is normally encrypted or deniably encrypted. The deniability of our scheme comes from the secret of the subgroup assignment, which is determined only once in the system setup phase. By the canceling property and the proper subgroup assignment, we can construct the released fake key to decrypt normal ciphertexts correctly.
- **Deterministic Decryption**. Most deniable encryption schemes have decryption error problems. These errors come from the designed decryption mechanisms. For example, in [12], Canetti et al. uses the subset decision mechanism for decryption. The receiver determines the decrypted message according to the subset decision result. If the sender chooses an element from the universal set but unfortunately the element is located in the specific subset, then an error occurs. The same error occurs in all translucent-set-based deniable encryption schemes. Another example is in [16], which uses a voting mechanism for decryption. Decryption is correct if and only if the correct part overwhelms the false part. Otherwise, the receiver will get the error result.
  The concept of our deniable scheme is different than these schemes described above. Our scheme extends a pairing ABE, which has a deterministic decryption algorithm, from the prime order group to the com-

3. The sender proof is still inter-independent, because receiver proofs are related to user keys whereas sender proofs are related to random choices for encryption.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

posite order group. The decryption algorithm in our scheme is still deterministic; therefore, there is no decryption errors using our scheme.

1.4 Organization

In additional to this introductory section, we introduce preliminaries used in this paper in Section 2. In Section 3, we formally define deniable CP-ABE and its properties. In Section 4, we show how to set up a basic deniable CP-ABE scheme and prove security, deniability and other features of our scheme. In Section 5, we transform our basic scheme from composite order groups to prime order groups. We then enhance our scheme to be chosen-ciphertext attack (CCA) secure in Section 6. In section 7, we implement our deniable schemes and evaluate their performance. Finally, we present our conclusions in Section 8.

2 Preliminaries

2.1 Prime Order Bilinear Groups

Let G and G_T be two multiplicative cyclic groups of prime order p, with map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let g be a generator of $G\mathbb{G}$. $\mathbb{G}$ is a bilinear map group if $\mathbb{G}$ and e have the following properties:

- Bilinearity: $\forall u, v \in G$ and $a, b \in Z$, $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$.
- Computability: the group action in $\mathbb{G}$ and map function e can be computed efficiently.

2.2 Waters CP-ABE

In this subsection, we provide an introduction to Waters CP-ABE [4]. Waters used LSSS to build an access control mechanism. Here, we first review the definition of LSSS.

Definition 1 (LSSS: Linear Secret Sharing Schemes [24]): A secret sharing scheme Π over set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

1) The shares for each party form a vector over $\mathbb{Z}_p$.
2) There exists a $l \times n$ matrix M called the share-generating matrix for Π. For all $i = 1, \ldots, l$, the i'th row of M is labeled by party $\rho(i)$, where ρ is a mapping function from $\{1, \ldots, l\}$ to party field $\mathcal{P}$. When considering column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, Mv is the vector of l shares of secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.

According to the above definition, an LSSS scheme has the linear reconstruction property. That is, given LSSS Π, access structure $\mathbb{A}$, and valid shares of a secret s, s can be recovered by those who have authorized sets. In [24], Beimel shows that the recovery procedure is time polynomial in the size of M. In an ABE scheme, parties represent attributes. The Waters CP-ABE scheme is composed of the following algorithms:

- **Setup**$() \rightarrow (MSK, PK)$: This algorithm chooses a bilinear group of prime order p with generator g, random elements $\alpha, a \in \mathbb{Z}_p$, and hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$. The public key PK is $\{g, e(g,g)^\alpha, g^a\}$ and the system secret key MSK is g^α.
- **Encrypt**$(PK, (M, \rho), \mathcal{M}) \rightarrow CT$: Given message $\mathcal{M}$ and LSSS access structure (M, ρ), this algorithm first chooses a random vector $\overrightarrow{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$. Let M be a $l \times n$ matrix and M_i denote the ith row of M. This algorithm calculates $\lambda_i = \overrightarrow{v} M_i, \forall i \in \{1, \ldots, l\}$. Further, this algorithm chooses $r_1, \ldots, r_l \in \mathbb{Z}_p$. The output ciphertext will be:

$$\begin{aligned} CT &= \{\mathcal{M} e(g,g)^{\alpha s}, g^s, (g^{a\lambda_1} H(\rho(1))^{-r_1}, g^{r_1}), \ldots, \\ &\quad (g^{a\lambda_l} H(\rho(l))^{-r_l}, g^{r_l})\} \\ &= \{C, C', (C_1, D_1), \ldots, (C_l, D_l)\}, \end{aligned}$$

  with a description of (M, ρ).
- **KeyGen**$(MSK, S) \rightarrow SK$: Given set S of attributes, this algorithm chooses $t \in \mathbb{Z}_p$ randomly and outputs the private key as:

$$K = g^{\alpha + at}, L = g^t, \forall x \in SK_x = H(x)^t.$$

- **Decrypt**$(CT, SK) \rightarrow \mathcal{M}$: Suppose that S satisfies the access structure and let $I \subset \{1, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. This algorithm finds a set of constants $\{w_i \in \mathbb{Z}_p\}$ such that $\sum_{i \in I} w_i \lambda_i = s$. The decryption algorithm computes

$$e(C', K) / (\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}) = e(g,g)^{\alpha s}$$

  and derives $\mathcal{M}$ from the ciphertext.

The security of Waters CP-ABE scheme is based on the decisional q-parallel bilinear BDHE assumption, which is defined as follows:

Definition 2 (Decisional q-parallel BDHE Assumption): Let $a, s, b_1, \ldots, b_q \xleftarrow{R} \mathbb{Z}_p$ and g be a generator of $\mathbb{G}$. Given

$$D := \left\{ \begin{array}{ll} g, g^s, g^a, \ldots, g^{(a^q)}, g^{(a^{q+2})}, \ldots, g^{(a^{2q})} & \\ \forall_{1 \leq j \leq q} & g^{s \cdot b_j}, g^{a/b_j}, \ldots, g^{(a^q/b_j)}, \\ & g^{(a^{q+2}/b_j)}, \ldots, g^{(a^{2q}/b_j)} \\ \forall_{1 \leq j,k \leq q, k \neq j} & g^{a \cdot s \cdot b_k / b_j}, \ldots, g^{a^q \cdot s \cdot b_k / b_j} \end{array} \right\}$$

and element $T \in \mathbb{G}_T$, we assume that for any PPT algorithm A that outputs in $\{0,1\}$,

$$Adv_A := |P[A(D, e(g,g)^{a^{q+1} s}) = 1] - P[A(D, T) = 1]|$$

is negligible.

Theorem 1: Suppose the decisional q-parallel BDHE assumption holds, then no polynomial time adversary can selectively break the Waters CP-ABE system in the CPA-model.

The proof can be found in [4] and is omitted here.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

2.3 Composite Order Bilinear Groups

The composite order bilinear group was first introduced in [25]; we use it to construct our scheme. Here we provide a brief introduction. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of composite order $N = p_1 p_2 \dots p_m$, where $p_1, p_2, \dots, p_m$ are distinct primes, with bilinear map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. For each prime p_i, $\mathbb{G}$ has a subgroup $\mathbb{G}_{p_i}$ of order p_i. We let $g_1, g_2, \dots, g_m$ be the generators of these subgroups respectively. Each element in $\mathbb{G}$ can be expressed in the form of $g_1^{a_1} g_2^{a_2} \dots g_m^{a_m}$, where $a_1, a_2, \dots, a_m \in \mathbb{Z}_N$. If a_i is congruent to zero modulo p_i, we say that this element has no $\mathbb{G}_{p_i}$ component. We say an element is in $\prod_{i \in S} \mathbb{G}_{p_i}$, where S is a subset from $1 \dots m$, if $\forall i \in S$, a_i is not congruent to zero modulo p_i.

The most important property of the composite bilinear groups is **orthogonality** between all subgroups under bilinear map e. This means that if $u \in \mathbb{G}_{p_i}, v \in \mathbb{G}_{p_j}$ and $i \neq j$, then $e(u, v) = 1$, where 1 is the identity element in $\mathbb{G}_T$.

The general complexity assumption used in the composite group is the **subgroup decision assumption**, stating that it is difficult to determine the existence of a given subgroup in a random composite order group element without orthogonality testing. The general form of this assumption is described as follows, as defined in [23]:

Definition 3 (General Subgroup Decision Assumption): Let $S_0, S_1, S_2, \dots, S_k$ be non-empty subsets of $1, \dots, m$ such that for each $2 \leq j \leq k$, either $S_j \cap S_0 = \emptyset = S_j \cap S_1$ or $S_j \cap S_0 \neq \emptyset \neq S_j \cap S_1$. Given group generator $\mathcal{G}$, we define the following distribution:

$$PP := \{N = p_1 p_2 \dots p_m, \mathbb{G}, \mathbb{G}_T, e\} \xleftarrow{R} \mathcal{G}$$
$$Z_i \xleftarrow{R} \mathbb{G}_{S_i} \forall i \in \{1, \dots, k\},$$
$$D := \{PP, Z_2, \dots, Z_k\}.$$

We assume that for that for any PPT algorithm A with output in $\{0, 1\}$,

$$Adv_{\mathcal{G},A} := |P[A(D, Z_0) = 1] - P[A(D, Z_1) = 1]|$$

is negligible.

This assumption also implies that it is hard to distinguish the outputs of the bilinear map function from other elements when they contain at least one common subgroup.

2.4 Chameleon Hash

The idea behind the chameleon hash scheme was first introduced in [26]. Just like other common secure hash functions, a chameleon hash scheme has two key properties, namely **collision resistance** and **semantic security**. Further, a chameleon hash scheme also provides **collision forgery** with a predetermined trapdoor. The input of a chameleon hash includes two parts, one being input message m and the other random string r. The random string r is used to provide a chance to adapt the message for the hash value. The definitions of the three aforementioned requirements, **collision resistance**, **semantic security** and **collision forgery**, are listed below.

Definition 4 (Collision Resistance): Given chameleon hash scheme $\{PK, SK, CH(\cdot, \cdot)\}$, where PK is the public information, SK is the trapdoor and $CH(\cdot, \cdot)$ is the hash function. Let m, m' be two different messages and r a random string. We call the scheme **collision resistant** if for any probabilistic polynomial time (PPT) algorithm A, it is hard to output r' such that $CH(m, r) = CH(m', r')$ without SK.

Definition 5 (Semantic Security): Given chameleon hash scheme $\{PK, SK, CH(\cdot, \cdot)\}$, where PK is the public information, SK is the trapdoor and $CH(\cdot, \cdot)$ is the hash function. We call the scheme **semantically secure** if for all pairs of message m, m' and random string r, the probability distribution of $CH(m, r)$ and $CH(m', r)$ are computationally indistinguishable.

Definition 6 (Collision Forgery): Given chameleon hash scheme $\{PK, SK, CH(\cdot, \cdot)\}$, where PK is the public information, SK is the trapdoor and $CH(\cdot, \cdot)$ is the hash function. Let m, m' be two different messages and r is a random string. We call the scheme a **collision forgery** scheme if there exists one PPT algorithm A that on input SK, outputs a string r' that satisfies $CH(m, r) = CH(m', r')$.

In this paper, we use CH to denote the chameleon hash public information and $CH(\cdot, \cdot)$ to denote the chameleon hash operation.

3 Definition

3.1 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 fake message 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 key-generation 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 being a normal set,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

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. IN [20], 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 blockwise deniable with a consistent encryption environment, we design our scheme to be a plan-ahead deniable encryption scheme. In [21], 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 plan-ahead, bi-deniable, and multi-distributional. Below, we provide the definition of this kind of deniable CP-ABE scheme.

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

- **Setup**$(1^\lambda) \rightarrow (PP, MSK)$: This algorithm takes security parameter λ as input and returns public parameter PP and system master key MSK.
- **KeyGen**$(MSK, S) \rightarrow SK$: Given set of attributes S and MSK, this algorithm outputs private key SK.
- **Enc**$(PP, \mathcal{M}, A) \rightarrow C$: This encryption algorithm takes as input public parameter PP, message $\mathcal{M}$, and LSSS access structure $A = (M, \rho)$ over the universe of attributes. This algorithm encrypts $\mathcal{M}$ and outputs a ciphertext C, which can be decrypted by those who possess an attribute set that satisfies access structure A. Note that A is contained in C.
- **Dec**$(PP, SK, C) \rightarrow \{\mathcal{M}, \perp\}$: This decryption algorithm takes as input public parameter PP, private key SK with its attribute set S, and ciphertext C with its access structure A. If S satisfies A, then this algorithm returns M; otherwise, this algorithm returns $\perp$.
- **OpenEnc**$(PP, C, \mathcal{M}) \rightarrow P_E$: This algorithm is for the sender to release encryption proof P_E for (M, C).
- **OpenDec**$(PP, SK, C, \mathcal{M}) \rightarrow P_D$: This algorithm is for the receiver to release decryption proof P_D for (M, C).
- **Verify**$(PP, C, M, P_E, P_D) \rightarrow \{T, F\}$: This algorithm is used to verify the correctness of P_E and P_D.
- **DenSetup**$(1^\lambda) \rightarrow (PP, MSK, PK)$: This algorithm takes security parameter λ as input and returns public parameters PP, system master key MSK, and system public key PK. PK is known by all system users and is kept secret to outsiders.
- **DenKeyGen**$(MSK, S) \rightarrow (SK, FK)$: Given set of attributes S and MSK, this algorithm outputs private key SK as well as FK for the user, where FK will be used for generating fake proof later.
- **DenEnc**$(PP, PK, \mathcal{M}, \mathcal{M}', A) \rightarrow C'$: Aside from the inputs of the normal encryption algorithm, this deniable encryption algorithm needs public key PK and fake message $\mathcal{M}'$. The output ciphertext must be indistinguishable from the output of **Enc**.
- **DenOpenEnc**$(PP, C', \mathcal{M}') \rightarrow P'_E$: This algorithm is for 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**$(PP, SK, FK, C', \mathcal{M}') \rightarrow P'_D$: This algorithm is for the receiver to release decryption proof P'_D for fake message $\mathcal{M}'$. The output must be indistinguishable from the result of **OpenDec** and must pass the **Verify** algorithm.

We require the following properties:

1) **Security**: The tuple {**Setup**,**KeyGen**,**Enc**,**Dec**} must form a secure CP-ABE scheme in a security model. In this work, we propose a CPA secure scheme and a CCA secure scheme. These two security models are defined in Section 3.2.
2) **Bi-deniability**: The CP-ABE is bi-deniable if, given public parameter PP, the two distribution tuples (M, C, P_E, P_D) and (M', C', P'_E, P'_D) are computational indistinguishable, where M, M' are claimed messages, C, C' are normally and deniably encrypted ciphertexts, respectively, and P_E, P_D, P'_E, P'_D are proofs generated from the normal and deniable open algorithms, respectively. That is, there is no PPT algorithm A for which

$$Adv_A := \left| \begin{array}{rl} & P[A(PP, (M, C, P_E, P_D)) = 1] \\ - & P[A(PP, (M', C', P'_E, P'_D)) = 1] \end{array} \right|$$

   is non-negligible.
3) **Deniable Receiver Proof Consistency**: The deniable CP-ABE is deniable receiver proof consistent if a deniable receiver proof is convincing even when considering all ciphertexts in the system. That is, given set of ciphertexts $\mathbb{C}$, including normally encrypted ciphertexts and deniably encrypted ciphertexts, normal proof P_D and deniable proof P'_D, there is no PPT algorithm A for which

$$Adv_A := |P[A(\mathbb{C}, P_D) = 1] - P[A(\mathbb{C}, P'_D) = 1]|$$

   is non-negligible.

We note that the last requirement is unusual for deniable encryption schemes. We build our scheme with this requirement for practicality. In a cloud storage service,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

it is impractical to frequently update security parameters. Therefore, coercers are able to check proofs with all stored encrypted files. For normal provided proofs, there will be no problems. So, our scheme must ensure deniable proofs to pass coercer checks, or coercers will know cheating has occurred. We also note that not all stored files are deniably encrypted. Some files are normally encrypted. A proposed receiver proof, regardless of whether it is normal or deniable, should be convincing for both normally and deniably encrypted files. We focus on receiver proofs instead of sender proofs because in most cases, senders add randomness during encryption. Therefore, any two sender proofs are usually independent, and sender proof consistency is unnecessary. For the above reasons, we build our scheme such that it adheres to the **Deniable Receiver Proof Consistency** requirement.

3.1.1 Is a Confidential PK Practical?

In the above definition, our scheme assumes that PK will be kept secret from the coercer. Some may argue that it is impractical, stating that coercers can pretend to be users in cloud storage services and obtain the PK. Once the PK is released to coercers, they can easily generate deniably encrypted ciphertexts and use these ciphertexts to determine the types of receiver proofs. To address this question, we must return to the basic assumption of deniable encryption schemes, i.e., **senders and receivers want to hide their communication messages from outside coercers**. Like all other cryptographic schemes, secrets must be assumed to be unknown to adversaries and our scheme is no exception. Therefore assuming that the PK is kept secret to coercers is acceptable and unavoidable.

To keep PK secret, cloud service providers can integrate deniable CP-ABE schemes with their own user authentication mechanisms. Note that in our definition, a deniable CP-ABE scheme can enable cloud storage service providers to offer two kinds of storage services, one being normal storage service, the other being audit-free storage service. So a user can choose to enjoy normal cloud storage services through a basic authentication process or enjoy audit-free cloud storage services through a much more sincere authentication process. Therefore, we believe our idea can be used to build practical cloud storage services, especially for those communities who currently have serious authentication processes.

3.2 Chosen-Plaintext-Attack (CPA) Security Model and Chosen-Ciphertext-Attack (CCA) Security Model

Here we describe the secure model for a CP-ABE scheme. An adversary is given a challenge question and is allowed to query an oracle for some information. The adversary wins the game if it can correctly answer the question. The formal security game is described as follows:

- **Setup:** The challenger first runs **Setup** and outputs PP to the adversary.
- **Phase 1:** The adversary generates queries $q_1, \ldots, q_m$ to the challenger. Query q_i can be one of the following two types of queries:
  - Key query: the adversary chooses attribute set S_i and obtains its private key from the challenger.
  - Decryption query: the adversary asks the challenger to decrypt ciphertext C_i and obtains its plaintext.
- **Challenge:** The adversary chooses two plaintexts M_0, M_1 for the challenger. The adversary also provides a challenge access structure A^*, which cannot be satisfied by the attributes used in $q_1, \ldots, q_m$. The challenger randomly chooses one bit $b \in \{0, 1\}$ and encrypts the message via $\mathbf{Enc}(PP, A^*, M_b) \rightarrow C^*$. The challenger sends C^* to the adversary as the challenge ciphertext.
- **Phase 2:** As in Phase 1, the adversary generates queries $q_{m+1}, \ldots, q_n$ to the challenger. Query q_i can be one of the following two types of queries:
  - Key query: the adversary chooses attribute set S_i and obtains its private key from the challenger. S_i cannot satisfy A^*.
  - Decryption query: the adversary asks the challenger to decrypt ciphertext C_i and obtains its plaintext. C_i cannot be C^*.
- **Guess:** The adversary returns guess result $b' \in \{0, 1\}$. The adversary wins if $b' = b$.

The advantage is defined as $|P(b' = b) - \frac{1}{2}|$.

Definition 8: A CP-ABE scheme is CPA secure if all polynomial time adversaries have at most a negligible advantage in the above game without any decryption queries.

Definition 9: A CP-ABE scheme is CCA secure if all polynomial time adversaries have at most a negligible advantage in the above game.

4 Deniable CP-ABE Construction

To build an audit-free secure cloud storage service, we use a deniable CP-ABE scheme as our core technology. We construct our basic deniable CP-ABE scheme, which is based on [4], as follows:

- **Setup**$(1^\lambda) \rightarrow (PP, MSK)$: This algorithm generates bilinear group $\mathbb{G}$ of order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are distinct primes with bilinear map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. $\mathbb{G}_T$ is also order N. We let $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ denote three orthogonal subgroups in $\mathbb{G}$ of order p_1, p_2, p_3, respectively. This algorithm then picks generators $g_1 \in \mathbb{G}_{p_1}, g_3 \in \mathbb{G}_{p_3}$, and randomly picks $\alpha, a \in \mathbb{Z}_N$. This algorithm also chooses hash function $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_{p_3}$. Public parameter PP is $\{\mathbb{G}, e, H_1, g_1 g_3, (g_1 g_3)^a, e(g_1 g_3, g_1 g_3)^\alpha\}$ and system secret key MSK is $(g_1 g_3)^\alpha$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

- **KeyGen**$(MSK, S) \rightarrow SK$: Given set S of attributes, this algorithm chooses $t \in \mathbb{Z}_N$ randomly and outputs private key SK as:

$$\begin{aligned} SK &= \{(g_1g_3)^{\alpha+at}, (g_1g_3)^t, \{H_1(x)^t\}_{\forall x \in S}\} \\ &= \{K, L, \{K_x\}_{\forall x \in S}\}. \end{aligned}$$

- **Enc**$(PP, \mathcal{M}, A = (M, \rho)) \rightarrow C$: Given message $\mathcal{M}$ and LSSS access structure (M, ρ). Let M be a $l \times n$ matrix and M_i denote the ith row of M. This algorithm first chooses two random vectors $\overrightarrow{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_N^n$ and $\overrightarrow{r} = (r_1, \ldots, r_l) \in \mathbb{Z}_N^l$. This algorithm then calculates $\lambda_i = \overrightarrow{v} M_i, \forall i \in \{1, \ldots, l\}$. In addition, this algorithm sets up one-way hash function $H(\cdot, \cdot)$[4] with two inputs. Note that hash function H can be any kind of one-way function and is determined during encryption. Each transaction may have different H. This algorithm flips two coins b_0, b_1 and picks two random string t_0, t_1. The output ciphertext C will be:

$$C = \{A_0, A_1, B, (C_1, D_1), \ldots, (C_l, D_l), H, t_0, t_1, V\},$$

where,

$$\begin{aligned} &A_{b_0} = \mathcal{M} \cdot e(g_1g_3, g_1g_3)^{\alpha s}, A_{1-b_0} \xleftarrow{R} \mathbb{G}_T, \\ &B = (g_1g_3)^s, \\ &C_i = (g_1g_3)^{a\lambda_i} H_1(\rho(i))^{-r_i}, D_i = (g_1g_3)^{r_i}, i = 1 \ldots l, \\ &V = H(\mathcal{M}, t_{b_1}) \neq H(A_{1-b_0} \cdot e(g_1g_3, g_1g_3)^{-\alpha s}, t_{1-b_1}). \end{aligned}$$

Access structure A is also attached to C.

- **Dec**$(PP, SK, C) \rightarrow \{\mathcal{M}, \perp\}$: To decrypt ciphertext C for access structure $A = (M, \rho)$, this algorithm first checks if attribute set S of SK satisfies A. Suppose S satisfies A and let $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then this algorithm finds a set of constants $\{w \in \mathbb{Z}_p\}$ such that $\sum_{i \in I} w_i \lambda_i = s$. This algorithm computes $\mathcal{M}_0, \mathcal{M}_1$ as follows:

$$M_{\{0,1\}} = A_{\{0,1\}} \cdot \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K)}$$

This algorithm then calculates

$$v_{i,j} = H(\mathcal{M}_i, t_j), \forall i, j \in \{0, 1\}.$$

If $v_{i,j}$ is equal to V, then $\mathcal{M}_i$ is the true message and is returned. Otherwise, this algorithm returns $\perp$.

- **OpenEnc**$(PP, C, \mathcal{M}) \rightarrow P_E$: This algorithm returns two coins b_0, b_1 as proof P_E.
- **OpenDec**$(PP, SK, C, \mathcal{M}) \rightarrow P_D$: This algorithm directly returns SK as proof P_D since this is the most persuasive proof.
- **Verify**$(PP, C, \mathcal{M}, P_E, P_D) \rightarrow \{T, F\}$: To verify P_E and P_D, this algorithm first runs **Dec**(PP, P_D, C) and checks if the output is equal to declared input $\mathcal{M}$. Then, this algorithm checks P_E with correct coins b_0, b_1 derived in the decryption process. If both requirements are satisfied, this algorithm returns T; otherwise, it returns F.
- **DenSetup**$(1^\lambda) \rightarrow (PP, MSK, PK)$: This algorithm runs **Setup**(1^λ) and obtains PP. System public key PK is $\{g_2g_3, (g_2g_3)^a, e(g_3, g_3)^\alpha, e(g_2g_3, g_2g_3)^\alpha\}$ and system secret key MSK is $\{(g_1g_3)^\alpha, g_1g_2g_3, (g_1g_2g_3)^\alpha\}$.
- **DenKeyGen**$(MSK, S) \rightarrow (SK, FK)$: This algorithm runs **KeyGen** and obtains SK for S. Next, this algorithm picks $t' \in \mathbb{Z}_N$ 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**$(PP, PK, \mathcal{M}, \mathcal{M}', A = (M, \rho)) \rightarrow C'$: This algorithm prepares $\lambda_i, \forall i \in \{1, \ldots, l\}$ just as the **Enc** algorithm does. This algorithm sets up chameleon hash function $CH(\cdot, \cdot)$. 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,

$$\begin{aligned} &A'_{b_0} = \mathcal{M} \cdot e(g_3, g_3)^{\alpha s}, A'_{1-b_0} = \mathcal{M}' \cdot e(g_2g_3, g_2g_3)^{\alpha s}, \\ &B' = (g_2g_3)^s, \\ &C'_i = (g_2g_3)^{a\lambda_i} H_1(\rho(i))^{-r_i}, D'_i = (g_1g_3)^{r_i}, i = 1, \ldots, l, \\ &V = CH(\mathcal{M}, t_{b_1}) = CH(\mathcal{M}', t_{1-b_1}). \end{aligned}$$

Based on the property of the chameleon hash, the sender can easily find t_{b_1} and t_{1-b_1} satisfying the above requirements.

- **DenOpenEnc**$(PP, C', \mathcal{M}') \rightarrow P'_E$: When the sender tries to fool the coercer with the pre-determined fake message, this algorithm returns two coins $1-b_1, 1-b_2$ as its proof P'_E.
- **DenOpenDec**$(PP, SK, FK, C', \mathcal{M}') \rightarrow P'_D$: This algorithm directly returns FK as proof P'_D.

4.1 Correctness

In this subsection, we show the correctness of this deniable CP-ABE scheme. There are four cases here:

1) When using ormal key SK to decrypt normally encrypted ciphertext C, the decryption process will be:

$$\begin{aligned} & \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K)} \\ = \; & \frac{\prod_{i \in I} (e((g_1g_3)^{a\lambda_i}, (g_1g_3)^t))^{w_i}}{e((g_1g_3)^s, (g_1g_3)^{\alpha+at})} \\ = \; & \frac{e(g_1g_3, g_1g_3)^{at \prod_{i \in I} \lambda_i w_i}}{e(g_1g_3, g_1g_3)^{s(\alpha+at)}} \\ = \; & e(g_1g_3, g_1g_3)^{-\alpha s}. \end{aligned}$$

4. We use H to represent a hash function's public information and $H(\cdot, \cdot)$ to represent the hash operation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

With the hash function in C and V, the receiver can derive message $\mathcal{M}$.

2) When using normal key SK to decrypt deniable ciphertext C' , the decryption process will be:

$$\begin{aligned} & \frac{\prod_{i\in I}(e(C'_i, L)e(D'_i, K_{\rho(i)}))^{w_i}}{e(B', K)} \\ = & \frac{\prod_{i\in I}(e((g_2g_3)^{a\lambda_i}, (g_1g_3)^t))^{w_i}}{e((g_2g_3)^s, (g_1g_3)^{\alpha+at})} \\ = & \frac{e(g_3, g_3)^{at\prod_{i\in I}\lambda_i w_i}}{e(g_3, g_3)^{s(\alpha+at)}} \\ = & e(g_3, g_3)^{-\alpha s}. \end{aligned}$$

With chameleon hash CH and V in C', the receiver can derive true message $\mathcal{M}$. Therefore, via the normal key, the receiver can obtain the correct message regardless of whether the message is normally encrypted.

3) When using deniable key FK to decrypt deniable ciphertext C', which is the case for fooling the coercer, the decryption process will be:

$$\begin{aligned} & \frac{\prod_{i\in I}(e(C'_i, L')e(D'_i, K'_{\rho(i)}))^{w_i}}{e(B', K')} \\ = & \frac{\prod_{i\in I}(e((g_2g_3)^{a\lambda_i}, (g_1g_2g_3)^t))^{w_i}}{e((g_2g_3)^s, (g_1g_2g_3)^{\alpha+at})} \\ = & \frac{e(g_2g_3, g_2g_3)^{at\prod_{i\in I}\lambda_i w_i}}{e(g_2g_3, g_2g_3)^{s(\alpha+at)}} \\ = & e(g_2g_3, g_2g_3)^{-\alpha s}. \end{aligned}$$

With chameleon hash CH and V in C', the receiver can derive fake message $\mathcal{M}'$. Therefore, the coercer will be convinced by $\mathcal{M}$ and FK.

4) When using deniable key FK to decrypt normal ciphertext C, which is the key compatible property, the decryption process will be:

$$\begin{aligned} & \frac{\prod_{i\in I}(e(C_i, L')e(D_i, K'_{\rho(i)}))^{w_i}}{e(B, K')} \\ = & \frac{\prod_{i\in I}(e((g_1g_3)^{a\lambda_i}, (g_1g_2g_3)^t))^{w_i}}{e((g_1g_3)^s, (g_1g_2g_3)^{\alpha+at})} \\ = & \frac{e(g_1g_3, g_1g_3)^{at\prod_{i\in I}\lambda_i w_i}}{e(g_1g_3, g_1g_3)^{s(\alpha+at)}} \\ = & e(g_1g_3, g_1g_3)^{-\alpha s}. \end{aligned}$$

Therefore, correct message $\mathcal{M}$ will be derived from normal ciphertext C, even though the key is deniable.

From the above, our scheme has two important properties. First, a user can obtain the true message with a valid secret key, regardless of whether the ciphertext is normally encrypted or deniably encrypted. Second, the fake key can be used to decrypt the normally encrypted ciphertext.

Theorem 2: Our CP-ABE system is receiver proof consistent.

Proof: In our scheme, we use keys as receiver proofs since keys are the most immediate proofs available. As shown above, both P_D and P'_D can be used to "correctly" decrypt these ciphertexts. By "correctly" here, we mean that a ciphertext can be decrypted to a meaningful message, which may be true or a pre-determined fake message. With P_D, regardless of whether a message is normally encrypted or deniably encrypted, the true message can be derived. As for P'_D, the decryption outputs are true messages when they are normally encrypted and are fake messages when true messages are deniably encrypted. Therefore, anyone who can differentiate between $(C_1, \ldots, C_n, P_D)$ and $(C_1, \ldots, C_n, P'_D)$ can also differentiate between true and pre-determined fake messages. In other words, these two tuples are indistinguishable. □

4.2 Security Proof

To prove that our deniable encryption scheme is secure requires this scheme to be a valid encryption scheme. For a multi-distributional deniable encryption scheme, it is only necessary to prove the security from the normal algorithm set. That is, we only need to prove the security of a scheme composed of the following four algorithms **Setup**, **KeyGen**, **Enc**, and **Dec**. As for the deniable algorithms, since deniable keys and ciphertexts are indistinguishable from normal keys and ciphertexts, which will be proved in the next subsection, deniable algorithms will be treated as normal algorithms which are proved to be secure. In other words, if the normal algorithm set can form a secure scheme, but the deniable set cannot, the security test will be a tool to distinguish these two sets of algorithms and there will be no deniability in our scheme. For proving security, we will reduce Waters CP-ABE to our deniable ABE scheme.

Theorem 3: Our proposed CP-ABE scheme is CPA secure if Waters CP-ABE is CPA secure.

Proof: Let $\mathcal{A}$ be an adversary that breaks the above deniable CP-ABE scheme. We can construct algorithm $\mathcal{B}$ that can break Waters CP-ABE as follows. $\mathcal{B}$ is given public parameters through the Waters CP-ABE scheme's **Setup** algorithm from challenger $\mathcal{X}$

$$PP_{p_3} := \{g_3, g_3^{a_3}, e(g_3, g_3)^{\alpha_3}\},$$

with prime number p_3, $\mathbb{G}_{p_3}$, $e(\cdot,\cdot)$ and $H_1(\cdot)$. For convenience, we use the suffix to represent different subgroups in our proof. Algorithm $\mathcal{B}$ proceeds as follows.

- **Setup:** $\mathcal{B}$ first picks two different prime numbers p_1 and p_2. Next, $\mathcal{B}$ generates group $\mathbb{G}$ with order $N = p_1p_2p_3$. Note that the subgroup with p_3 order in $\mathbb{G}$ should be the same as $\mathbb{G}_{p_3}$. $\mathcal{B}$ sets up PP_{p_1} with the Waters CP-ABE **Setup** algorithm from $\mathbb{G}_{p_1}$ and outputs $\{g_1, g_1^{a_1}, e(g_1, g_1)^{\alpha_1}\}$, where a_1, α_1 are in $\mathbb{Z}_{p_1}$. Next, $\mathcal{B}$ shows

$$PP := \{g_1g_3, g_1^{a_1}g_3^{a_3}, e(g_1, g_1)^{\alpha_1}e(g_3, g_3)^{\alpha_3}\}$$

to A with N, $\mathbb{G}_N$, $e(\cdot,\cdot)$ and $H_1(\cdot)$. Note that $e(\cdot,\cdot)$ and $H_1(\cdot)$ are the same with the given function from $\mathcal{X}$. Though a_3 is secret and different from a_1,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

which comes from $\mathbb{Z}_{p_3}$ and $\mathbb{Z}_{p_1}$ respectively, $g_1^{a_1} g_3^{a_3}$ can be treated as $(g_1 g_3)^a$, where $a \in \mathbb{Z}_N$ from the Chinese remainder theorem. For the same reason, $e(g_1, g_1)^{\alpha_1} e(g_3, g_3)^{\alpha_3}$ can be treated as $e(g_1 g_3, g_1 g_3)^{\alpha}$, where $\alpha \in \mathbb{Z}_N$

- **Phase 1:** When $\mathcal{B}$ receives a key generation query for attribute set S from $\mathcal{A}$, $\mathcal{B}$ simply relays the query to $\mathcal{X}$ and obtains $SK_{p_3} = \{K_{p_3}, L_{p_3}, \{K_x\}_{\forall x \in S}\}$. $\mathcal{B}$ generates K_{p_1}, L_{p_1} with the same algorithm. Next, $\mathcal{B}$ replies $\mathcal{A}$ the secret key SK as follows:

$$SK = \{K_{p_1} K_{p_3}, L_{p_1} L_{p_3}, \{K_x\}_{\forall x \in S}\}.$$

- **Challenge:** $\mathcal{A}$ outputs two messages $\mathcal{M}_0$, $\mathcal{M}_1$ with access structure (M, ρ) to $\mathcal{B}$. $\mathcal{B}$ directly relays $\mathcal{M}_0$, $\mathcal{M}_1$ and (M, ρ) to $\mathcal{X}$ as the challenge and obtains

$$\{\mathcal{M}^* \cdot e(g_3, g_3)^{\alpha_3 s_3}, B_{p_3}, (C_{1,p_3}, D_{1,p_3}), \ldots, (C_{l,p_3}, D_{l,p_3})\}$$

from $\mathcal{X}$. $\mathcal{M}^* \in \{\mathcal{M}_0, \mathcal{M}_1\}$ is chosen by $\mathcal{X}$. $\mathcal{B}$ setups a chameleon hash function CH and randomly picks b_1, b_2 from $\{0, 1\}$, $\{r_1, \ldots, r_l\}, s_1$ from $\mathbb{Z}_{p_1}$. $\mathcal{B}$ also calculates $\{\lambda'_1, \ldots, \lambda'_l\}$. Finally, $\mathcal{B}$ outputs C to $\mathcal{A}$ as follows:

$$C = \{A_0, A_1, B, (C_1, D_1), \ldots, (C_l, D_l), CH, t_0, t_1, V\},$$

where

$$A_{b_1} = \mathcal{M}^* \cdot e(g_1, g_1)^{\alpha_1 s_1} e(g_3, g_3)^{\alpha_3 s_3}, A_{1-b_1} \xleftarrow{R} \mathbb{G}_N,$$
$$B = B_{p_3} \cdot g_1^{s_1},$$
$$C_i = C_{i,p_3} \cdot g_1^{a_1 \lambda'_i}, D_i = D_{i,p_3} g_1^{r'_i}, \forall i \in \{1, \ldots, l\},$$
$$V = CH(\mathcal{M}_0, t_{b_2}) = CH(\mathcal{M}_1, t_{1-b_2}).$$

Because of the Chinese remainder theorem, $\mathcal{A}$ will treat C as a ciphertext that comes from secret $s \in \mathbb{Z}_N$. Here, a chameleon hash function is used instead of a normal hash function; however, to $\mathcal{A}$, who has no trapdoor for the chameleon hash function, the chameleon hash function is just a normal one-way hash function.

- **Phase 2:** $\mathcal{A}$ submits key generation queries to $\mathcal{B}$ and $\mathcal{B}$ responds as shown in Phase 1.
- **Guess:** Finally, adversary $\mathcal{A}$ outputs guess b' to $\mathcal{B}$ and $\mathcal{B}$ uses b' to reply $\mathcal{X}$.

If $\mathcal{A}$ achieves a non-negligible advantage against the deniable scheme from our construction, $\mathcal{B}$ can use the output of $\mathcal{A}$ to also achieve a non-negligible advantage against the Waters ABE scheme in the CPA model. □

Combined with Theorem 1, we have the following theorem:

Theorem 4: Our proposed CP-ABE scheme is CPA secure if the q-BDHI assumption holds.

4.3 Deniability Proof

To prove the deniability of our CP-ABE scheme, we must show (M, C, P_E, P_D) and (M', C', P'_E, P'_D) are indistinguishable. Since M,C,P_E,P_D are pairwise independent because of the security property, we need only show the indistinguishability between C and C', P_E and P'_E, and P_D and P'_D.

Lemma 1: Under the general subgroup decision assumption, normal ciphertext C and deniable ciphertext C' are indistinguishable.

Proof: We suppose there exists PPT attacker $\mathcal{A}$ who achieves a non-negligible advantage in distinguishing the deniable ciphertext from the normal ciphertext of our scheme. We can create PPT algorithm $\mathcal{B}$ that has a non-negligible advantage against the general subgroup decision assumption.

$\mathcal{B}$ receives $N = p_1 p_2 p_3, g_1 g_3, g_2 g_3, T$, where g_1, g_2, g_3 belong to $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$, $\mathbb{G}_{p_3}$ respectively. $\mathcal{B}$ wants to know if T belongs to G_{p_1,p_3} or G_{p_2,p_3}. $\mathcal{B}$ runs the **DenSetup**(1^λ) algorithm and obtains PP, PK, and MSK. $\mathcal{B}$ then releases PP to $\mathcal{A}$. Note that PK is unnecessary because the coercer should not obtain the PK information. $\mathcal{A}$ sends key queries to $\mathcal{B}$ with attribute sets and $\mathcal{B}$ replies to all queries with normal keys via the **KeyGen** algorithm. In the challenge phase, $\mathcal{B}$ receives an encryption challenge from $\mathcal{A}$ with an access structure and two messages $\mathcal{M}$ and $\mathcal{M}'$. The access structure cannot be satisfied by any attribute sets that have been queried. $\mathcal{B}$ flips two coins b_0, b_1 and returns the following ciphertext to $\mathcal{A}$:

$$C = \{A_0, A_1, B, (C_1, D_1), \ldots, (C_l, D_l), CH, t_0, t_1, V\},$$

where,

$$A_{b_0} = \mathcal{M} \cdot e(T, g_1 g_3)^{\alpha}, A_{1-b_0} = \mathcal{M}' \cdot e(T, g_2 g_3)^{\alpha},$$
$$B = T,$$
$$C_i = T^{a M_{i,1}} H(\rho(i))^{-r_i}, D_i = (g_1 g_3)^{r_i}, i = 1, \ldots, l,$$
$$V = CH(\mathcal{M}, t_{b_1}) = CH(\mathcal{M}', t_{1-b_1}).$$

$M_{i,1}$ is the first element in the ith row of M. If $T = (g_1 g_3)^s \in G_{p_1,p_3}$, then C is a normal ciphertext; if $T = (g_2 g_3)^s \in G_{p_2,p_3}$, then C is a deniable ciphertext. $\mathcal{A}$ can still send key queries to $\mathcal{B}$ and receive normal secret keys. Finally $\mathcal{A}$ answers $\mathcal{B}$ if the ciphertext is deniable. If $\mathcal{A}$ has a non-negligible advantage over the ciphertext decision problem, $\mathcal{B}$ can also have a non-negligible advantage over the subgroup decision problem. □

Lemma 2: Normal encryption proof P_E and deniable encryption proof P'_E are indistinguishable.

Proof: Since the encryption proof is composed of two random coins, P_E and P'_E are indistinguishable. We note that given ciphertext C, it is impossible to build a PPT algorithm that can correctly find P_E with a non-negligible advantage because of the security property. □

Lemma 3: Under the general subgroup decision assumption, normal decryption proof P_D and deniable decryption proof P'_D are indistinguishable.

Proof: In this scheme, we use private key SK as the decryption proof. Therefore, this lemma is equal to the indistinguishability of SK and FK. The only difference between SK and FK is the existence of element g_2. That is, the key decision problem is a subgroup decision

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

problem, which is hard according to the general subgroup decision assumption. Therefore, P_D and P'_D are indistinguishable. □

From the three lemmas above, we yield the following conclusion:

Theorem 5: Under the general subgroup decision assumption, our CP-ABE system is bi-deniable.

4.4 Decryption Errors

In section 1, we described why most deniable schemes may cause decryption errors. Most of these schemes claim their decryption error rates are small or negligible, but they cannot ensure that there are no errors whatsoever in their schemes. In our scheme, a receiver uses a one-way function with a signature to obtain the true message. Both the one-way function and the signature are generated by the sender. That is, the sender can avoid any decryption errors in encryption.

5 Deniable CP-ABE Construction from Prime Order Bilinear Group

In the previous section, we described how to design a deniable CP-ABE scheme with composite order bilinear groups for building audit-free cloud storage services. Composite order bilinear groups have two attractive properties, namely **projecting** and **cancelling**, defined by Freeman in [22]. We make use of the cancelling property for building a consistent environment; however, Freeman also pointed out the important problem of computational cost in regard to the composite order bilinear group. The bilinear map operation of a composite order bilinear group is much slower than the operation of a prime order bilinear group with the same security level. That is, in our scheme, a user will spend too much time in decryption when accessing files on the cloud. To make composite order bilinear group schemes more practical, Freeman converted [25], [27], and [28] into prime order schemes. Meiklejohn et al. showed that both projecting and cancelling cannot be simultaneously achieved in prime order groups in [29].

For the same reason, we use a simulating tool proposed by Lewko in [23] to convert our composite order bilinear group scheme to a prime order bilinear group scheme. This tool is based on dual orthonormal bases and the subspace assumption. Different subgroups are simulated as different orthonormal bases and therefore, by the orthogonal property, the bilinear operation will be cancelled between different subgroups. Our formal deniable CP-ABE construction method uses only the cancelling property of the composite order group. Therefore, Lewko's tool will be suitable with our construction. The converting process is straightforward. First, we generate dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*)$ of $\mathbb{Z}_p^n$, where p is the prime order and n is the dimension. Each subgroup in the public parameter has its own basis vector in $\mathbb{D}$ and uses the relative basis vector in $\mathbb{D}^*$ when generating keys. This step is slightly different from Lewko's system; Lewko uses more than one basis to simulate one subgroup because multiple key elements are combined into one. We simply make all key elements separate and therefore only use one basis for one subgroup. Before explaining our construction, we present some notation below.

- For $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$ and $g \in \mathbb{G}$, we use $g^{\mathbf{v}}$ to denote n-tuple of elements $(g^{v_1}, \ldots, g^{v_n})$.
- We use e_n to denote the product of the componentwise pairings:

$$e_n(g^{\mathbf{v}}, g^{\mathbf{w}}) = \prod_{i=1}^{n} e(g^{v_i}, g^{w_i}) = e(g, g)^{\mathbf{v} \cdot \mathbf{w}}.$$

We next describe how to simulate our scheme with prime order bilinear groups.

- **Setup**$(1^\lambda) \rightarrow (PP, MSK)$: This algorithm generates bilinear group $\mathbb{G}$ of prime order p with bilinear map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. $\mathbb{G}_T$ is also of order p. This algorithm generates dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*)$ from $\mathbb{Z}_p^3$. Let $\mathbb{D} = (\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$ and $\mathbb{D}^* = (\mathbf{d}_1^*, \mathbf{d}_2^*, \mathbf{d}_3^*)$. We then have the following property:

$$\begin{cases} \mathbf{d}_i \cdot \mathbf{d}_j^* = 0 (\text{mod } p), i \neq j \\ \mathbf{d}_i \cdot \mathbf{d}_j^* = \psi (\text{mod } p), i = j \end{cases}.$$

  The algorithm then picks generator $g \in \mathbb{G}$ and $\alpha, \gamma, a \in \mathbb{Z}_p$. The algorithm also chooses hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_p$. Public parameter PP is $\{\mathbb{G}, e, H_1, g^{\mathbf{d}_1}, g^{a\mathbf{d}_1}, g^{\mathbf{d}_3}, g^{a\mathbf{d}_3}, \mathbf{d}_3, e(g,g)^{\psi(\alpha+\gamma)}\}$ and system secret key MSK is $\{g^{\mathbf{d}_1^*}, g^{a\mathbf{d}_1^*}, g^{\alpha\mathbf{d}_1^*}, g^{\mathbf{d}_3^*}, g^{a\mathbf{d}_3^*}, g^{\gamma\mathbf{d}_3^*}, \mathbf{d}_3^*\}$.
- **KeyGen**$(MSK, S) \rightarrow SK$: Given set S of attributes, this algorithm chooses $t \in \mathbb{Z}_p$ randomly and outputs private key SK as:

$$\begin{aligned} SK &= \left\{ \begin{array}{l} g^{(\alpha+at)\mathbf{d}_1^* + (\gamma+at)\mathbf{d}_3^*}, g^{t(\mathbf{d}_1^* + \mathbf{d}_3^*)}, \\ \{H_1(x)^{t\mathbf{d}_3^*}\}_{\forall x \in S} \end{array} \right\} \\ &= \{K, L, \{K_x\}_{\forall x \in S}\}. \end{aligned}$$

- **Enc**$(PP, \mathcal{M}, A = (M, \rho)) \rightarrow C$: Given message $\mathcal{M}$ and LSSS access structure (M, ρ). Let M be a $l \times n$ matrix and M_i denote the ith row of M. This algorithm first chooses two random vectors $\overrightarrow{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$ and $\overrightarrow{r} = (r_1, \ldots, r_l) \in \mathbb{Z}_p^l$. This algorithm then calculates $\lambda_i = \overrightarrow{v} M_i, \forall i \in \{1, \ldots, l\}$. In addition, the algorithm sets up a one-way hash function $H(\cdot, \cdot)$ with two inputs. Note that hash function H can be any one-way function and is determined during encryption. Each transaction may have different H. This algorithm flips two coins b_0, b_1 and picks two random string t_0, t_1. Output ciphertext C will be:

$$C = \{A_0, A_1, B, (C_1, D_1), \ldots, (C_l, D_l), H, t_0, t_1, V\},$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

where,

$A_{b_0} = \mathcal{M} \cdot e(g,g)^{s\psi(\alpha+\gamma)}, A_{1-b_0} \xleftarrow{R} \mathbb{G}_T$,
$B = g^{s(\mathbf{d}_1+\mathbf{d}_3)}$,
$C_i = g^{a\lambda_i(\mathbf{d}_1+\mathbf{d}_3)} H_1(\rho(i))^{-r_i\mathbf{d}_3}, i = 1 \dots l$,
$D_i = g^{r_i(\mathbf{d}_1+\mathbf{d}_3)}, i = 1 \dots l$,
$V = H(\mathcal{M}, t_{b_1}) \neq H(A_{1-b_0} \cdot e(g,g)^{-s\psi(\alpha+\gamma)}, t_{1-b_1})$.

Access structure A is also attached to C.

- **Dec**$(PP, SK, C) \rightarrow \{\mathcal{M}, \perp\}$: To decrypt ciphertext C for access structure $A = (M, \rho)$, this algorithm first checks if attribute set S of SK satisfies A. Suppose S satisfies A and let $I \subset \{1, 2, \dots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, this algorithm finds a set of constants $\{w \in \mathbb{Z}_p\}$ such that $\sum_{i \in I} w_i \lambda_i = s$. The algorithm computes $\mathcal{M}_0, \mathcal{M}_1$ as follows:

$$M_{\{0,1\}} = A_{\{0,1\}} \cdot \frac{\prod_{i \in I}(e(C_i, L)e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K)}$$

This algorithm then calculates

$$v_{i,j} = H(\mathcal{M}_i, t_j), \forall i, j \in \{0, 1\}.$$

If $v_{i,j}$ is equal to V, then $\mathcal{M}_i$ is the true message and is returned. Otherwise, this algorithm returns $\perp$.

- **DenSetup**$(1^\lambda) \rightarrow (PP, MSK, PK)$: This algorithm runs **Setup**(1^λ) and obtains PP. This algorithm randomly picks $\beta \in \mathbb{Z}_p$. System public key PK is $\{g^{\mathbf{d}_2}, g^{a\mathbf{d}_2}, e(g,g)^{\psi\gamma}, e(g,g)^{\psi\beta}\}$ and system secret key MSK is $\{g^{\mathbf{d}_1^*}, g^{a\mathbf{d}_1^*}, g^{\alpha\mathbf{d}_1^*}, g^{\mathbf{d}_2^*}, g^{a\mathbf{d}_2^*}, g^{\beta\mathbf{d}_2^*}, g^{\mathbf{d}_3^*}, g^{a\mathbf{d}_3^*}, g^{\gamma\mathbf{d}_3^*}, \mathbf{d}_3^*\}$.
- **DenKeyGen**$(MSK, S) \rightarrow (SK, FK)$: This algorithm runs **KeyGen** and obtains SK for S. Then, the algorithm picks $t' \in \mathbb{Z}_p$ and generates FK as follows:

$$\begin{aligned} FK &= \left\{ \begin{array}{l} g^{(\alpha+at')\mathbf{d}_1^*+(\beta+at')\mathbf{d}_2^*+(\gamma+at')\mathbf{d}_3^*}, \\ g^{t'(\mathbf{d}_1^*+\mathbf{d}_2^*+\mathbf{d}_3^*)}, \{H_1(x)^{t'\mathbf{d}_3^*}\}_{\forall x \in S} \end{array} \right\} \\ &= \{K', L', \{K'_x\}_{\forall x \in S}\}. \end{aligned}$$

- **DenEnc**$(PP, PK, \mathcal{M}, \mathcal{M}', A = (M, \rho)) \rightarrow C'$: This algorithm prepares $\lambda_i, \forall i \in \{1, \dots, l\}$ as the **Enc** algorithm does. The algorithm sets up chameleon hash function $CH(\cdot, \cdot)$. 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:

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

where,

$A'_{b_0} = \mathcal{M} \cdot e(g,g)^{s\psi\gamma}, A'_{1-b_0} = \mathcal{M}' \cdot e(g,g)^{s\psi(\beta+\gamma)}$,
$B' = g^{s(\mathbf{d}_2+\mathbf{d}_3)}$,
$C'_i = g^{a\lambda_i(\mathbf{d}_2+\mathbf{d}_3)} H_1(\rho(i))^{-r_i\mathbf{d}_3}, i = 1 \dots l$,
$D'_i = g^{r_i(\mathbf{d}_1+\mathbf{d}_3)}, i = 1 \dots l$,
$V = CH(\mathcal{M}, t_{b_1}) = CH(\mathcal{M}', t_{1-b_1})$.

Based on the property of the chameleon hash, the sender can easily find t_{b_1} and t_{1-b_1} to satisfy the above requirements.

- **DenOpenEnc**, **DenOpenDec**, **Verify**, **DenOpenEnc**, **DenOpenDec** are the same as our basic scheme and therefore not described here.

By this construction, we make different bases form different subgroups. According to Definition 5 in [23], this approach follows the subgroup decision assumption. Therefore, bi-deniability also holds in this construction.

6 CCA Secure Deniable CP-ABE Scheme

In [30], Boneh et al. proved that an IND-sID-CPA secure IBE scheme can be transformed into an IND-sID-CCA secure scheme with the help of one-time signature scheme ($\mathcal{G}$, **Sign**, **Verify**). The one-time signature is used to maintain the integrity of the ciphertext. Using the same technique, we can enhance our CPA secure deniable CP-ABE scheme to be a CCA secure deniable CP-ABE scheme, as demonstrated in [31]. We modify the following algorithms for this enhancement:

- **Setup**$_{\text{cca}}(1^\lambda) \rightarrow (PP_{\text{cca}}, MSK)$: Aside from the original **Setup** algorithm process, this algorithm additionally chooses hash function $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_{p_3}$ and randomly picks $b \in \mathbb{Z}_N$. This algorithm attaches $(g_1 g_3)^b, H_2$ to public parameter PP from the original **Setup** algorithm as PP_{cca}.
- **Enc**$_{\text{cca}}(PP_{\text{cca}}, \mathcal{M}, A) \rightarrow C_{\text{cca}}$: The sender first runs the original **Enc** algorithm and obtains $C = \{A_0, A_1, B, (C_1, D_1), \dots, (C_l, D_l), H, t_0, t_1, V\}$. The sender generates $B_2 = (g_1 g_3)^{bs}$. The output C_{cca} will be

$$C_{\text{cca}} = \left\{ \begin{array}{l} A_0, A_1, B, B_2, \\ (C_1, D_1), \dots, (C_l, D_l), \\ H, t_0, t_1, V, V_2 \end{array} \right\},$$

where

$$V_2 = H_2 \left(\begin{array}{l} A_0, A_1, B, B_2, (C_1, D_1), \dots, \\ (C_l, D_l), H, t_0, t_1, V \end{array} \right)^s.$$

- **Dec**$_{\text{cca}}(PP_{\text{cca}}, SK, C_{\text{cca}}) \rightarrow M$: The receiver first verifies the following two equations:

$$e(B, (g_1 g_3)^b) \stackrel{?}{=} e(B_2, g_1 g_3),$$

$$V_3 = H_2 \left(\begin{array}{l} A_0, A_1, B, B_2, (C_1, D_1), \dots, \\ (C_l, D_l), H, t_0, t_1, V \end{array} \right),$$

$$e(V_3, B_2) \stackrel{?}{=} e(V_2, (g_1 g_3)^b).$$

If the above two equations do not hold, this algorithm returns $\perp$. Otherwise, we proceed as the original algorithm.

- **DenSetup**$_{\text{cca}}(1^\lambda) \rightarrow (PP_{\text{cca}}, MSK, PK_{\text{cca}})$: This algorithm executes **Setup**$_{\text{cca}}$ and obtains PP_{cca}, MSK. The algorithm also generates $(g_2 g_3)^b$. PK_{cca} is PK, which is derived from **DenSetup**, and $(g_2 g_3)^b$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

- **DenEnc**$_{\text{cca}}(PP_{\text{cca}}, PK_{\text{cca}}, \mathcal{M}, \mathcal{M}', A) \rightarrow C'_{\text{cca}}$: The sender first runs the original **DenEnc** algorithm and obtains $C' = \{A'_0, A'_1, B', (C'_1, D'_1), \ldots, (C'_l, D'_l), CH, t_0, t_1, V\}$. The sender generates $B'_2 = (g_2 g_3)^{bs}$. The output C_{cca} will be

$$C'_{\text{cca}} = \left\{ \begin{array}{l} A'_0, A'_1, B', B'_2, \\ (C'_1, D'_1), \ldots, (C'_l, D'_l), \\ CH, t_0, t_1, V, V_2 \end{array} \right\},$$

where

$$V_2 = H_2 \left(\begin{array}{l} A'_0, A'_1, B', B'_2, (C'_1, D'_1), \ldots, \\ (C'_l, D'_l), CH, t_0, t_1, V \end{array} \right)^s.$$

Theorem 6: Our enhanced scheme is CCA secure if our basic scheme is CPA secure.

Proof: The difference between the CPA and CCA models is that CCA allows the existence of a decryption oracle. Therefore, we focus here on how to answer the adversary's decryption queries. When receiving a decryption query, the oracle proceeds as follows:

1) If $e(B, (g_1 g_3)^b) = e(B_2, g_1 g_3)$ and $e(V_3, B_2) = e(V_2, (g_1 g_3)^b)$ do not hold, return $\perp$.
2) In phase 2, if both equations hold and the queried ciphertext is the same as the challenged ciphertext, return $\perp$.
3) The oracle generates SK for a set S that can satisfy the access structure in the ciphertext and decrypt the ciphertext. Then, the oracle returns the decryption result.

□

From theorem 4 and theorem 6, we have the following conclusion:

Theorem 7: Our enhanced scheme is CCA secure if the q-BDHI assumption holds.

7 Performance Evaluation

In this section, we evaluate the performance of our idea by implementing two deniable schemes: the composite order scheme and the prime order simulation scheme. We compare them with the Waters scheme [4]. We use the Pairing Based Cryptography (PBC) library for cryptographic operations. We use type A1 pairing because this type of pairing can support both prime order and composite order groups. In our experiment, we set the size of each prime to 512 bits, which is equal to 256 bits of security [32]. Under this setting, the composite group order size is 1536 bits. However, when considering security, the composite order scheme with a group size of 1536 bits is equal to the prime order scheme with a group size of 512 bits. This is because a message is encrypted in one subgroup whose group size is 512 bits.

Our experiments focus on encryption and decryption performance. The **Setup** and **KeyGen** performance are skipped because these two algorithms are not time critical. The four **Open** algorithms are low-cost algorithms because these algorithms only return existing information. The cost of **Verify** algorithm is equal to that of **Dec**. Note that we do not distinguish deniable encryption from normal encryption; their numbers of arithmetic operations and pairing operations are equal, and therefore the normal one and the deniable one will have similar performance. In our design, the encryption cost and the decryption cost depend on required attribute numbers. For convenience, we make all attributes mandatory as our cryptographic policy. We run the experiments with different attribute numbers, from 10 to 1000. Our experiments focus on one block encryption/decryption. Each block is set to 128 bytes because PBC reads around 130 bytes to generate a $\mathbb{G}_T$ element when the group size is 512 bits[5]. A large file can be divided into multiple blocks, and all blocks can be protected by one secret s. Because $\mathbb{G}_T$ multiplication and H are lightweight operations, we use one-block encryption/decryption to evaluate the performance. The experiments are tested on a virtual machine with 3.47 GHz CPU and 8 GB memory.

Figures 1 and 2 show the experiment results. As we can see, encryption time and decryption time grow linearly over the attribute number in all three schemes. The composite order scheme is undoubtedly the most time-consuming scheme; its performance is almost unacceptable for practical applications. The reason for this poor

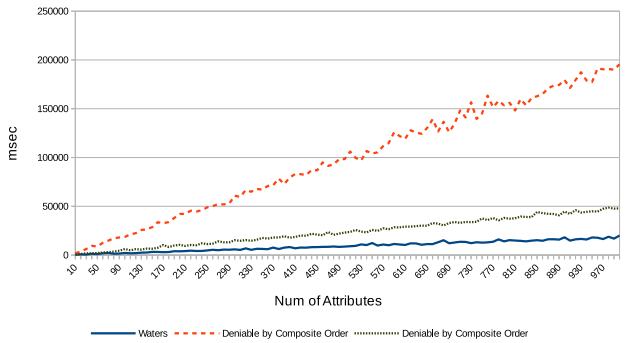

Fig. 1. Encryption benchmark.

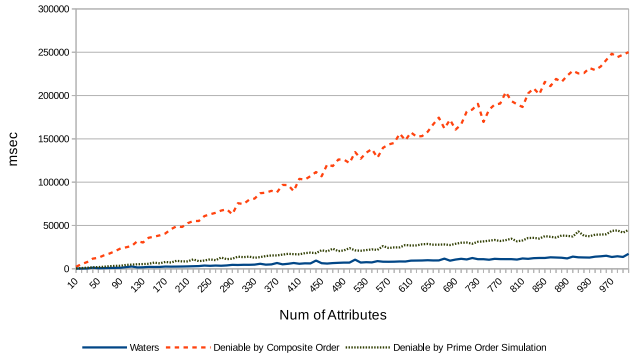

Fig. 2. Decryption benchmark.

5. In the composite order scheme, the group size is 1536 bits and PBC reads around 388 bytes to generate a $\mathbb{G}_T$ element. For simplicity, our experiments fix a block size to 128 bytes in three schemes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2015.2424882, IEEE Transactions on Cloud Computing

performance is that all arithmetic and pairing operations are executed in a group much larger than those for the other two schemes. As for the prime order simulation scheme, it takes little time to get the deniability feature from the Waters scheme and therefore, the prime order simulation scheme is suitable to be distributed to cloud storage services for the deniability feature.

8 Conclusions

In this work, we proposed a deniable CP-ABE scheme to build an audit-free cloud storage service. The deniability feature makes coercion invalid, and the ABE property ensures secure cloud data sharing with a fine-grained access control mechanism. Our proposed scheme provides a possible way to fight against immoral interference with the right of privacy. We hope more schemes can be created to protect cloud user privacy.

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 and Privacy*, 2007, pp. 321–334.

[4] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography*, 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.

[6] S. Hohenberger and B. Waters, "Attribute-based encryption with fast decryption," in *Public Key Cryptography*, 2013, pp. 162–179.

[7] P. K. Tysowski and M. A. Hasan, "Hybrid attribute- and re-encryption-based key management for secure and scalable mobile applications in clouds." *IEEE T. Cloud Computing*, pp. 172–186, 2013.

[8] Wired. (2014) Spam suspect uses google docs; fbi happy. [Online]. Available: http://www.wired.com/2010/04/cloud-warrant/

[9] Wikipedia. (2014) Global surveillance disclosures (2013present). [Online]. Available: http://en.wikipedia.org/wiki/Global_surveillance_disclosures_(2013-present)

[10] ——. (2014) Edward snowden. [Online]. Available: http://en.wikipedia.org/wiki/Edward_Snowden

[11] ——. (2014) Lavabit. [Online]. Available: http://en.wikipedia.org/wiki/Lavabit

[12] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable encryption," in *Crypto*, 1997, pp. 90–104.

[13] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Eurocrypt*, 2010, pp. 62–91.

[14] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theor. Comput. Sci.*, vol. 422, pp. 15–38, 2012.

[15] M. Dürmuth and D. M. Freeman, "Deniable encryption with negligible detection probability: An interactive construction," in *Eurocrypt*, 2011, pp. 610–626.

[16] A. O'Neill, C. Peikert, and B. Waters, "Bi-deniable public-key encryption," in *Crypto*, 2011, pp. 525–542.

[17] P. Gasti, G. Ateniese, and M. Blanton, "Deniable cloud storage: sharing files via public-key deniability," in *WPES*, 2010, pp. 31–42.

[18] M. Klonowski, P. Kubiak, and M. Kutylowski, "Practical deniable encryption," in *SOFSEM*, 2008, pp. 599–609.

[19] M. H. Ibrahim, "A method for obtaining deniable public-key encryption," *I. J. Network Security*, vol. 8, no. 1, pp. 1–9, 2009.

[20] J. B. Nielsen, "Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case," in *Crypto*, 2002, pp. 111–126.

[21] R. Bendlin, J. B. Nielsen, P. S. Nordholt, and C. Orlandi, "Lower and upper bounds for deniable public-key encryption," Cryptology ePrint Archive, Report 2011/046, 2011, http://eprint.iacr.org/.

[22] D. M. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups," in *Eurocrypt*, 2010, pp. 44–61.

[23] A. B. Lewko, "Tools for simulating features of composite order bilinear groups in the prime order setting," in *Eurocrypt*, 2012, pp. 318–335.

[24] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Israel Institute of technology, 1996.

[25] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *TCC*, 2005, pp. 325–341.

[26] H. Krawczyk and T. Rabin, "Chameleon signatures," in *NDSS*, 2000.

[27] D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *Eurocrypt*, 2006, pp. 573–592.

[28] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Eurocrypt*, 2008, pp. 146–162.

[29] S. Meiklejohn, H. Shacham, and D. M. Freeman, "Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures," in *Asiacrypt*, 2010, pp. 519–538.

[30] D. Boneh, R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," *SIAM J. Comput.*, vol. 36, no. 5, pp. 1301–1328, 2007.

[31] K. Liang, L. Fang, D. S. Wong, and W. Susilo, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," *IACR Cryptology ePrint Archive*, vol. 2013, p. 236, 2013.

[32] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management: Part 1: General (revision 3)," NIST, Tech. Rep., 2012.

Po-Wen Chi received his B.S. and M.S. in Electrical Engineering from National Taiwan University in 2003 and 2005. He is currently a Ph.D. candidate in the Department of Electrical Engineering at National Taiwan University. His research interests include network security, applied cryptography, software-defined networking, and telecommunications.

Chin-Laung Lei received the B.S. degree in Electrical Engineering from the National Taiwan University, Taipei, in 1980 and the Ph.D. degree in computer science from the University of Texas at Austin in 1986. From 1986 to 1988, he was an assistant professor in the Computer and Information Science Department, Ohio State University, Columbus. In 1988, he joined the faculty of the Department of Electrical Engineering, National Taiwan University, where he is now a professor. He is a cowinner of the first IEEE LICS test-of-time award, and has published more than 250 technical articles in scientific journals and conference proceedings. His current research interests include network security, cloud computing, and multimedia QoE management.