Generic and Efficient Constructions of Attribute-Based Encryption with Verifiable Outsourced Decryption

Xianping Mao, Junzuo Lai, Qixiang Mei, Kefei Chen, and Jian Weng

Abstract—Attribute-based encryption (ABE) provides a mechanism for complex access control over encrypted data. However in most ABE systems, the ciphertext size and the decryption overhead, which grow with the complexity of the access policy, are becoming critical barriers in applications running on resource-limited devices. Outsourcing decryption of ABE ciphertexts to a powerful third party is a reasonable manner to solve this problem. Since the third party is usually believed to be untrusted, the security requirements of ABE with outsourced decryption should include privacy and verifiability. Namely, any adversary including the third party should learn nothing about the encrypted message, and the correctness of the outsourced decryption is supposed to be verified efficiently. We propose generic constructions of CPA-secure and RCCA-secure ABE systems with verifiable outsourced decryption from CPA-secure ABE with outsourced decryption, respectively. We also instantiate our CPA-secure construction in the standard model and then show an implementation of this instantiation. The experimental results show that, compared with the existing scheme, our CPA-secure construction has more compact ciphertext and less computational costs. Moreover, the techniques involved in the RCCA-secure construction can be applied in generally constructing CCA-secure ABE, which we believe to be of independent interest.

Index Terms—Attribute-based encryption, outsourced decryption, verifiability, RCCA

1 Introduction

Cloud computing is a very fascinating computing paradigm, in which computation and storage are moved away from terminal devices to the cloud. This new and popular paradigm brings important revolutions and makes bold innovations for the manner in which enterprises and individuals manage, distribute, and share content. By outsourcing their information technology capabilities to some cloud service providers, cloud users may achieve significant cost savings.

There is widespread public concern about cloud computing, that is, how to ensure cloud users' data security. Part of the outsourced data is sensitive, and should only be accessed by authorized data consumers at remote locations. For instance, in a university, one of its colleges uploads its (encrypted) development projects to the university cloud, and wants to give only the administrative personnel of the university and the faculty of this college the privilege to access the encrypted projects. This is a very natural scenario in our real life when we use cloud storage systems. Cryptographic technology is an essential manner to achieve this goal. For example, attribute-based encryption (ABE) [1], a special kind of powerful functional encryptions [2], contributes to access control over encrypted data.

The notion of ABE was first introduced by Sahai and Waters [1]. Depending on how to deploy the access control policy, there are two different kinds of ABE systems. That is, key-policy attribute-based encryption (KP-ABE) [3] and its dual notion, ciphertext-policy attribute-based encryption (CP-ABE) [4]. In a CP-ABE scheme, every ciphertext is associated with an access policy, and every user's private key is associated with a set of attributes. While in a KP-ABE scheme, ciphertexts are labeled with sets of attributes and access policies over these attributes are associated with users' private keys. In an ABE system, decryption operation requires that the set of attributes should match the access policy. Given its expressiveness, ABE is regarded as one of the most natural and important technologies for realizing data access control in the cloud.

However, in most existing ABE schemes, one of the main efficiency drawbacks is that the size of the ciphertext and the decryption overhead (computational cost) grow with the complexity of the access policy. This becomes critical barriers in applications running on resource-limited devices. For instance, the college adopts a pairing-based ABE scheme to encrypt its development projects and uploads the generated ABE ciphertext to the university cloud. An authorized administrative officer of the university, who is on a business ship, wants to look up the encrypted development projects of the college through his (resource-limited) mobile phone. He then wants to download and decrypt the ABE ciphertext. Since the ciphertext might have a large size and the pairing operations in decryption procedure are usually

- *X. Mao is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: maoxp@sjtu.edu.cn.*
- *J. Lai and J. Weng are with the Department of Computer Science, Jinan University, Guangzhou 510632, China. E-mail: {junzuolai, cryptjweng}@gmail.com.*
- *Q. Mei is with the College of Information, Guangdong Ocean University, Zhanjiang 524088, China. E-mail: nupf@163.com.*
- *K. Chen is with the School of Science, Hangzhou Normal University, Hangzhou 310036, China. E-mail: kfchen@hznu.edu.cn.*

Manuscript received 7 Jan. 2015; revised 16 Mar. 2015; accepted 16 Mar. 2015. Date of publication 16 Apr. 2015; date of current version 2 Sept. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TDSC.2015.2423669

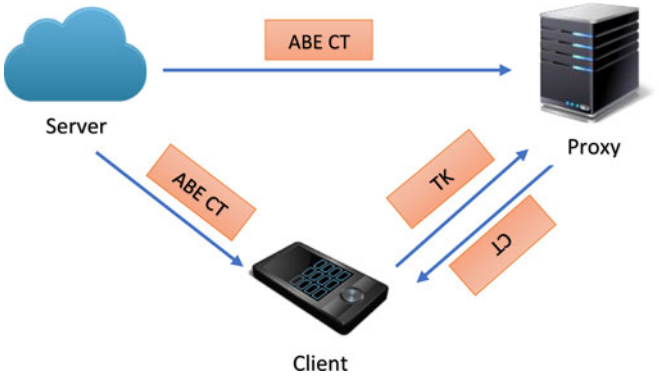


Fig. 1. ABE system with outsourced decryption.

expensive for a resource-limited device, he has to wait for a long time and sometimes even aborts the decryption procedure.

To remarkably eliminate the ciphertext size and the decryption overhead for users in ABE systems, a new method for efficiently and securely outsourcing decryption of ABE ciphertexts was put forth by Green et al. [5]. Fig. 1 illustrates their ABE system with outsourced decryption. More concretely, in their ABE system with outsourced decryption, the key generation algorithm is modified to produce two keys for a user. The first one is a short El Gamal type secret key, called the retrieving key rk, and it must be kept private by the user. The second one is the corresponding transformation key tk, which is shared with a proxy and can be publicly distributed. Essentially, this key pair is generated using a key blinding technique [5]. When the proxy receives a CP-ABE (resp. KP-ABE) ciphertext for an access policy (resp. attributes set) for which the attributes set (resp. access policy) associated with the user satisfies, it is able to use the transformation key to convert the ABE ciphertext into a simple and short El Gamal-style ciphertext which decrypts to the same message. After receiving this short and partially-decrypted ciphertext, the user is then able to decrypt this simple ciphertext with one exponentiation. Based on the existing ABE schemes, Green et al. proposed two ABE schemes with outsourced decryption, which are selectively secure under the chosen-plaintext attacks (selectively CPA-secure). Furthermore, by using the Fujisaki-Okamoto transformation [6], these two selectively CPA-secure ABE schemes with outsourced decryption are converted into two selectively RCCA-secure schemes in the random oracle model.

The ABE system with outsourced decryption might run into some problems: the ill-disposed proxy wants to save computing cost, the proxy is under malicious attack or other system malfunctions. In these situations, the data receiver might want to verify that the outsourced transformation of the (valid) normal ABE ciphertext was indeed done correctly. In [5], Green et al. also informally discussed the verifiability of the proxy's transformation and provided a method to verify the correctness of the outsourcing decryption in the random oracle model. Since the random oracle model is heuristic and a proof of security in the random oracle model does not directly imply anything about the security in the real world [7], Lai et al. managed to provide a verifiability mechanism which does not rely on random oracles [8]. They first refined the syntax of ABE with outsourced decryption to allow for the verifiability of the transformations, and then proposed a formal security notion of verifiability of ABE with outsourced decryption. Afterwards, based on Waters' (small-universe) CP-ABE scheme [9], they constructed a concrete CP-ABE scheme with verifiable outsourced decryption, which is selectively CPA-secure in the standard model. Generally speaking, they added to the ciphertext an encryption of an extra random message and a checksum value, which is computed with this random message and the actual plaintext. This checksum value plays a role of the commitment to the actual plaintext and is used to check whether the transformation is indeed done correctly.

However, the technique adopted in [8] inevitably brings on some negative side-effects. On one hand, the encryption of an extra random message doubles the computational overhead of both the data sender and the proxy. On the other hand, the normal (i.e., non-transformed) ciphertext size is twice as long as that of the underlying original ABE scheme (namely Waters' CP-ABE scheme, used as a core building block). This increases the bandwidth requirement and computational costs of resource-limited devices.

Accordingly, it is meaningful to find out a more efficient manner to provide the verifiability of outsourced decryption for ABE systems with outsourced decryption, especially in the standard model. Another interesting question is how to construct a RCCA-secure ABE scheme with outsourced decryption, which also supports the verifiability of outsourced decryption, in the standard model.

1.1 Our Contribution

In this paper, we give affirmative responses to the two problems mentioned above. We shall give *generic* constructions of CPA-secure and RCCA-secure ABE with *verifiable* outsourced decryption from CPA-secure ABE with outsourced decryption, respectively. At a very high level, both of our generic constructions have a "decrypt-then-verify" flavor. That is, after receiving a partially-decrypted ciphertext transformed by a proxy, the data receiver first decrypts it using some simple operations and then verifies the correctness of the outsourced decryption.

In our CPA-secure construction, unlike the technique of separately encrypting an extra random message and then using this random message to commit to the true message in [8], our method is encrypting a message and a random value together and then committing to the message by using the random value. Our construction is generic and has smaller ciphertext size and less computational costs. It also makes us obtain generic RCCA-secure construction more naturally.

To achieve RCCA-security, we first bring the properties of ciphertext verifiability and delegatability in ABE systems, proposed by Yamada et al. [10], to ABE systems with outsourced decryption. We show that if a CPA-secure ABE scheme with outsourced decryption has ciphertext verifiability or delegatability, it then can be transformed into a RCCA-secure ABE scheme with *verifiable* outsourced decryption by combining a secure encapsulation scheme, a strong one-time message authentication code and a secure commitment scheme.

Our constructions can be also applied to selectively CPA-secure ABE systems with outsourced decryption, which will result in selectively CPA-secure/RCCA-secure ABE systems with verifiable outsourced decryption. Moreover, since an ABE scheme with outsourced decryption just adds transformation procedures to a traditional ABE scheme in essence, it is a remarkable fact that the techniques involved in our RCCA-secure transformation can be directly applied in generally constructing CCA-secure ABE from CPA-secure ABE, which we believe to be of independent interest.

By instantiating our generic CPA-secure construction based on Green et al.'s (small-universe and backwards-compatible variant of) slectively CPA-secure CP-ABE system with outsourced decryption [5] and Pedersen Commitment [11], we obtain a selectively CPA-secure CP-ABE system with *verifiable* outsourced decryption in the standard model. Note that Green et al.'s CP-ABE scheme with outsourced decryption [5] has both ciphertext verifiability and delegatability. By combining the Boneh-Katz encapsulation scheme [12], the CBC-MAC using AES as the underlying block cipher [12] and Pedersen Commitment [11], it can be transformed into a selectively RCCA-secure ABE system with *verifiable* outsourced decryption in the *standard* model.

To evaluate our ABE systems with verifiable outsourced decryption, we implemented our selectively CPA-secure CP-ABE instantiation and Lai et al.'s scheme based on Charm [13], and conducted experiments on both an ARM-based smartphone and an Intel-core desktop computer, which are modeled as a resourced-limited device and a powerful proxy, respectively. The experimental results show that, compared with Lai et al.'s system, our selectively CPA-secure system has more compact ciphertext and less computational costs. Another observation is that outsourcing the decryption operations from the resource-limited smartphone to the powerful computer can significantly reduce the computational costs of the smartphone.

1.2 Our Technique

At a high level, in order to construct such ABE schemes with verifiable outsourced decryption, we begin with a CPA-secure ABE scheme with outsourced decryption and adopt the "decrypt-then-verify" paradigm.

In the generic construction of CPA-secure ABE scheme with verifiable outsourced decryption, we employ a commitment scheme to verify the correctness of outsourced decryption. More concretely speaking, to encrypt a message m in some proper message space, the data sender first selects a random value r. Then, the data sender uses the underlying ABE scheme with outsourced decryption to encrypt $m\|r$ to ct' and commits to m using the value r to get a commitment cm. The final (normal) ciphertext is $\langle ct', cm\rangle$. After receiving the partially-decrypted ciphertext pct transformed by a proxy using the transformation key, the data receiver first decrypts the partially-decrypted ciphertext pct to $m\|r$ by using the corresponding retrieving key, runs the revealing algorithm of the commitment scheme to verify the correctness of the transformation and then decides whether to accept message m as the correct decryption result.

Inspired by [12] and [10], in the generic construction of RCCA-secure ABE scheme with verifiable outsourced decryption, besides a commitment scheme, we also employ an encapsulation scheme and a massage authentication code. More concretely speaking, to encrypt a message m in some proper message space, the data sender first uses the encapsulation algorithm of the underlying encapsulation scheme to generate a random value r, an encapsulation string com and a decapsulation string dec, where r can be split into two equal-length strings r_1 and r_2. Here we assume that $r = r_1\|r_2$. Then, the data sender encrypts $m\|dec$ to generate a ciphertext component ct' using the underlying CPA-secure ABE scheme with outsourced decryption. Subsequently, the data sender generates a message authentication code tag on the ABE ciphertext ct' using r_1 as the authentication key and generates a commitment cm to m using r_2 as the auxiliary information. The final (normal) ciphertext is $\langle com, ct', tag, cm\rangle$. After receiving the partially-decrypted ciphertext pct transformed by a proxy using the transformation key, the data receiver first decrypts the partially-decrypted ciphertext pct to $m\|dec$ by using the retrieving key, and then recovers r from com and dec using the decapsulation algorithm. Lastly, the data receiver verifies the message authentication code and opens the commitment to check message m. In this RCCA-secure construction, it is required that the underlying CPA-secure ABE scheme with outsourced decryption should have ciphertext verifiability or delegatability in order to answer decryption queries.

1.3 Related Work

After Sahai and Waters' seminal work [1], Goyal et al. [3] and Bethencourt et al. [4] proposed the first KP-ABE scheme and the first CP-ABE scheme, respectively. These two ABE schemes support any monotone access structures. Subsequently, a great number of ABE schemes have been proposed, such as [14], [15], [16], [17], [18] and [9]. In [15], Cheung et al. presented a CP-ABE scheme which is proved to be secure in the standard model. Lewko et al. [16] proposed a fully secure CP-ABE scheme. In [9], Waters proposed several very efficient CP-ABE constructions. These constructions work for any access policy that can be expressed in terms of a linear secret sharing scheme (LSSS) matrix. Ostrovsky et al. [14] proposed a KP-ABE scheme that allows a user's private key to be expressed in terms of any access formula over attributes. Okamoto and Takashima [18] proposed KP-ABE and CP-ABE schemes with non-monotone access structures. A large amount of work employs ABE to realize fine-grained data access control, such as [19], [20], [21], and [22]. Chase [23] proposed multi-authority ABE which allows any polynomial number of independent authorities to monitor attributes and issue secret keys. His proposal uses the concepts of a trusted central authority and global identifiers. Chow and Chase [24] (somewhat) removed the trusted central authority by using a distributed pseudorandom function. Lekwo and Waters [25] proposed a new decentralized multi-authority ABE system.

It is very necessary to speed up the decryption operations of ABE systems, especially on some resource-limited devices. The pairing delegation techniques proposed in [26] can enable users to outsource the pairing computations to a third party. But using the pairing delegation techniques in the decryption operations of ABE system, the drawback that computational cost is proportional to the size of the

access policy still exists. To speed up the decryption of ABE systems, there are several trade-offs in terms of efficiency and security. The decryption algorithms in [27] and [28] require a constant number of pairing computations, while the security of their constructions can be proved only in the selective model. Green et al. [5] proposed a new paradigm for ABE systems to reduce the decryption time for resource-limited devices with the help of a powerful third party. Besides outsourcing decryption of ABE systems, outsourcing of key-issuing for ABE systems is also considered in [29] and [30]. Outsourcing encryption for ABE systems is another considerable direction [31].

As we know, the third party is untrusted, one needs to verify the correctness of outsourced decryption. The techniques proposed in [32] and [33] can be used to outsource decryption in ABE systems, since they deal with outsourcing of general computation problems and provide both privacy and verifiability for the outsourcing procedure. But because both their constructions are based on fully homomorphic encryption systems proposed by Gentry [34], these solutions are far away from being practical now [35]. Lai et al. [8] provided a practical method, which is used to verify the correctness of outsourced decryption in their ABE system.

2 Preliminaries

We begin by defining some notations that will be used throughout this paper. Let λ be a security parameter, which implies that all other quantity parameters are polynomial functions of λ. Denote by $\mathbb{Z}$ the integer and denote by $\mathbb{Z}_p$ the set of integers modulo p. For a positive integer k, denote by $[k]$ the set $\{1,\ldots,k\}$. A function f is negligible if for every polynomial $p(\cdot)$, there exists an N_0 such that for all integers $n > N_0$ it holds that $f(n) < 1/p(n)$. For a probabilistic algorithm $\mathcal{A}$, let $y \leftarrow \mathcal{A}(x;R)$ denote the process of running $\mathcal{A}$ on input x and inner randomness $R \in \mathcal{R}_{\mathcal{A}}$ and assigning y the result, where $\mathcal{R}_{\mathcal{A}}$ is the randomness space of $\mathcal{A}$. We write $y \leftarrow \mathcal{A}(x)$ for $y \leftarrow \mathcal{A}(x;R)$ with R chosen from $\mathcal{R}_{\mathcal{A}}$ uniformly at random. If $\mathcal{S}$ is a finite set, $y \leftarrow \mathcal{S}$ denotes that y is chosen from $\mathcal{S}$ uniformly at random. The symbol $\|$ denotes string concatenation.

2.1 Bilinear Groups

Let $\mathcal{G}$ be an algorithm that takes as input a security parameter λ and outputs a quadruple $(p,\mathbb{G},\mathbb{G}_{\mathrm{T}},e)$, where $\mathbb{G}$ and $\mathbb{G}_{\mathrm{T}}$ are (multiplicative) cyclic groups of prime order p such that the group action in $\mathbb{G}$ and $\mathbb{G}_{\mathrm{T}}$ can be computed efficiently. The bilinear map $e:\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{G}_{\mathrm{T}}$ fulfills the following three properties:

Bilinearity. For all $g_1,g_2\in\mathbb{G}$ and $a,b\in\mathbb{Z}_p$, we have $e(g_1^a,g_2^b)=e(g_1,g_2)^{ab}$.

Non-degeneracy. $e(g_1,g_2)\neq 1_{\mathbb{G}_{\mathrm{T}}}$, whenever $g_1,g_2\neq 1_{\mathbb{G}}$.

Efficiency. The bilinear map e can be efficiently computed.

We refer to the quadruple $(p,\mathbb{G},\mathbb{G}_{\mathrm{T}},e)$ as a bilinear group and the algorithm $\mathcal{G}(1^\lambda)$ as a bilinear group generator.

2.2 Commitment and Encapsulation

A commitment scheme is a two-phase protocol between two participants, a sender and a receiver. Here we only consider so-called non-interactive commitment scheme, in which all the communication goes from the sender to the receiver. A non-interactive commitment scheme $\mathcal{CS}=(\mathsf{InitCom},\mathsf{Commit},\mathsf{Reveal})$ with message space $\mathcal{M}$, randomness space $\mathcal{V}$ and commitment space $\mathcal{D}$ consists of the following three algorithms:

$\mathsf{InitCom}(1^\lambda)$ The initialization algorithm takes as input a security parameter λ. It outputs the (public) commitment parameters cp. When it is clear from the context, we omit cp in the input of the following algorithms.

$\mathsf{Commit}(m,r)$ The commitment algorithm takes as input a message $m\in\mathcal{M}$ and auxiliary information $r\in\mathcal{V}$. It outputs a commitment $cm\in\mathcal{D}$.

$\mathsf{Reveal}(m,r,cm)$ The revealing algorithm takes as input a message m, auxiliary information r and a commitment cm. It outputs a bit $b\in\{0,1\}$ to indicate whether cm is a commitment to m, where $b=1$ signifies "acceptance" and $b=0$ signifies "rejection".

The correctness of commitment scheme requires that, for all cp output by $\mathsf{InitCom}$ and for any message $m\in\mathcal{M}$ and any auxiliary information $r\in\mathcal{V}$, we have $\mathsf{Reveal}(m,r,\mathsf{Commit}(m,r))=1$.

Say a commitment scheme is secure if it satisfies two security properties, namely the hiding and binding properties. Informally, the hiding property says that a commitment cm to m does not reveal any information about m, whereas the binding property says that a commitment cm to m cannot be opened to another value m'. Here, we only require that both hiding and binding hold computationally. We define these two properties formally by two experiments $\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$, respectively.

Experiment $\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$:
$cp\leftarrow\mathsf{InitCom}(1^\lambda), r\in\mathcal{R}$
$m_0,m_1\leftarrow\mathcal{A}, \beta\leftarrow\{0,1\}$
$cm\leftarrow\mathsf{Commit}(m_\beta,r)$
$\beta'\leftarrow\mathcal{A}(\lambda,cp,cm)$
If $\beta=\beta'$, then return 1,
else return 0.

Experiment $\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$:
$cp\leftarrow\mathsf{InitCom}(1^\lambda), r\in\mathcal{R}, m\in\mathcal{M}$
$cm\leftarrow\mathsf{Commit}(m,r)$
$(m',r')\leftarrow\mathcal{A}(\lambda,cp,m,r,cm)$
If $\mathsf{Reveal}(m',r',cm)=1$,
then return 1,
else return 0.

The advantage of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$ is defined as $\mathsf{Adv}^{\mathrm{Hiding}}_{\mathsf{CS},\mathcal{A}}(\lambda)=|\Pr[\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{CS},\mathcal{A}}(\lambda)]-\frac{1}{2}|$, and the advantage of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$ is defined as $\mathsf{Adv}^{\mathrm{Binding}}_{\mathsf{CS},\mathcal{A}}(\lambda)=|\Pr[\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{CS},\mathcal{A}}(\lambda)]=1|$. A commitment scheme is secure if both $\mathsf{Adv}^{\mathrm{Hiding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathrm{Binding}}_{\mathsf{CS},\mathcal{A}}(\lambda)$ are negligible (in the security parameter λ).

Encapsulation [12] may be viewed as a weak variant of commitment. An encapsulation scheme commits the sender to a random string as opposed to a message chosen by the sender as in the case of commitment. In term of security, it only requires binding to hold for honestly-generated encapsulations. An encapsulation scheme $\mathcal{ES}=(\mathsf{Init},\mathsf{Encap},\mathsf{Decap})$ with randomness space $\mathcal{V}$, encapsulation string space $\mathcal{E}$ and decapsulation string space $\mathcal{D}$ consists of the following three PPT algorithms:

$\mathsf{Init}(1^\lambda)$ On input a security parameter λ, the initialization algorithm outputs public parameters pub.

$\mathsf{Encap}(pub)$ On input the public parameters pub, the encapsulation algorithm outputs (r,com,dec), where $r\in\mathcal{V}$,

$com \in \mathcal{E}$ and $dec \in \mathcal{D}$. com and dec are referred to as the encapsulation string and the decapsulation string, respectively.

$\mathsf{Decap}(pub, com, dec)$ On input the public parameters pub, an encapsulation string com and a decapsulation string dec, the decapsulation algorithm outputs $r \in \mathcal{V} \cup \{\perp\}$.

The correctness of encapsulation scheme requires that, for all pub output by Init and for all (r, com, dec) output by $\mathsf{Encap}(pub)$, we have $\mathsf{Decap}(pub, com, dec) = r$. For simplicity, we also assume that r, com and dec have fixed lengths for any given value of the security parameter λ.

As in the case of commitment, an encapsulation scheme satisfies the security notions of binding and hiding. Similarly, both hiding and binding are required to hold only computationally. Consider the following two experiments $\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$.

Experiment $\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$:
$(pub) \leftarrow \mathsf{Init}(1^\lambda)$
$(r_1, com, dec) \leftarrow \mathsf{Encap}(pub)$
$r_0 \leftarrow \mathcal{V}$, $\beta \leftarrow \{0,1\}$
$\beta' \leftarrow \mathcal{A}(\lambda, pub, com, r_\beta)$
If $\beta = \beta'$, then return 1,
else return 0.

Experiment $\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$:
$(pub) \leftarrow \mathsf{Init}(1^\lambda)$
$(r, com, dec) \leftarrow \mathsf{Encap}(pub)$
$dec' \leftarrow \mathcal{A}(\lambda, pub, com, dec)$
If $\mathsf{Decap}(pub, com, dec') \notin \{r, \perp\}$,
then return 1,
else return 0.

The advantage of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$ is defined as $\mathsf{Adv}^{\mathrm{Hiding}}_{\mathsf{ES},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathrm{Hiding}}_{\mathsf{ES},\mathcal{A}}(\lambda)] - \frac{1}{2}|$, and the advantage of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$ is defined as $\mathsf{Adv}^{\mathrm{Binding}}_{\mathsf{ES},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathrm{Binding}}_{\mathsf{ES},\mathcal{A}}(\lambda)] = 1|$. An encapsulation scheme is secure if both $\mathsf{Adv}^{\mathrm{Hiding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathrm{Binding}}_{\mathsf{ES},\mathcal{A}}(\lambda)$ are negligible (in the security parameter λ).

2.3 Massage Authentication Code

A message authentication code (MAC) $\mathcal{MA} = (\mathsf{Mac}, \mathsf{Ver})$ with the message space $\mathcal{M}$, the key space $\mathcal{K}$ and the tag space $\mathcal{T}$ consists of the following two PPT algorithms [12]:

$\mathsf{Mac}_{sk}(m)$ On input a key $sk \in \mathcal{K}$ and a message $m \in \mathcal{M}$, the tagging algorithm outputs a tag $tag \in \mathcal{T}$.

$\mathsf{Ver}_{sk}(m, tag)$ On input a key sk, a message m and a tag tag, the verification algorithm outputs a bit $b \in \{0,1\}$, where $b = 1$ signifies "acceptance" and $b = 0$ signifies "rejection".

The correctness of MAC requires that for every key $sk \in \mathcal{K}$, every message $m \in \mathcal{M}$, and all tag output by $\mathsf{Mac}_{sk}(m)$, we have $\mathsf{Ver}_{sk}(m, tag) = 1$.

We consider the secuirty notion of strong one-time message authentication code. This security notion is defined using a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. First, both $\mathcal{A}$ and $\mathcal{C}$ are given λ as input. Then, $\mathcal{C}$ chooses a random key $sk \in \mathcal{K}$ and $\mathcal{A}$ may issue at most one tagging query on m^*. $\mathcal{C}$ responds with $tag^* = \mathsf{Mac}_{sk}(m^*)$. Finally, $\mathcal{A}$ outputs (m, tag). $\mathcal{A}$ wins the game if $\mathsf{Ver}_{sk}(m, tag) = 1$ but $(m, tag) \neq (m^*, tag^*)$. Note that $\mathcal{A}$ may win even if $m = m^*$. The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}^{\mathrm{SOT}}_{\mathsf{MA},\mathcal{A}}(\lambda) = |\Pr\ [\mathcal{A} \text{ wins}]|$. Say a message authentication code is a strong one-time message authentication code if any PPT adversary has at most a negligible advantage in the above game.

2.4 ABE with (Verifiable) Outsourced Decryption

Let $\mathcal{S}$ denote a set of attributes and let $\mathbb{A}$ denote an access structure. First we define a Boolean function R which takes as input an attribute set $\mathcal{S}$ and an access structure $\mathbb{A}$ and outputs a bit in $\{0,1\}$. If the attributes set $\mathcal{S}$ satisfies the access structure $\mathbb{A}$, the function R outputs 1, otherwise it outputs 0. We consider two distinct varieties of ABE with (verifiable) outsourced decryption: ciphertext-policy and key-policy. For general description, we will define (I_{key}, I_{enc}) as the input to the key generation and encryption algorithm respectively. Hence, in a CP-ABE scheme with outsourced decryption, we have $(I_{key}, I_{enc}) = (\mathcal{S}, \mathbb{A})$. While in a KP-ABE scheme with outsourced decryption, we have $(I_{key}, I_{enc}) = (\mathbb{A}, \mathcal{S})$.

In the original model defined in [5], a transformation key tk and a corresponding retrieving key rk are created by the master authority, and the input to the outsourcing decryption algorithm only includes the user's retrieving key and the partially-decrypted ciphertext, but does not include the normal ciphertext. If a malicious proxy returns to the user another valid partially-decrypted ciphertext (e.g, a partially-decrypted ciphertext the proxy once generated honestly), the user cannot detect this malicious behavior [8]. Lai et al. [8] refined the model to achieve the verifiability. In their model, the user generates the transformation key from his private key by himself. It is more flexible since the user can determine whenever he wants to outsource decryption. In addition, the input to algorithm $\mathsf{Decrypt}_{out}$ includes the normal ciphertext and the partially-decrypted ciphertext. It is not difficult to see that Green et al.'s ABE schemes with outsourced decryption [5] can be transformed into ABE schemes with outsourced decryption in Lai et al.'s model. Here we adopt Lai el al.'s refined model. Formally, a CP-ABE (resp. KP-ABE) scheme with outsourced decryption consists of the following seven algorithms [8]:

$\mathsf{Setup}(1^\lambda, U)$. The system setup algorithm takes as input a security parameter λ and a description of attribute universe U. It outputs the public parameters $params$ and a master secret key msk. In the input of all the following algorithms, for convenient notation, we omit $params$ which is implicitly included.

$\mathsf{KeyGen}(msk, I_{key})$. The key generation algorithm takes as input the master security key msk and an attribute set (resp. access structure) I_{key}. It outputs a private key $sk_{I_{key}}$.

$\mathsf{Encrypt}(m, I_{enc})$. The encryption algorithm takes as input a message m and an access structure (resp. attribute set) I_{enc}. It outputs a normal ciphertext ct.

$\mathsf{Decrypt}(sk_{I_{key}}, ct)$. The decryption algorithm takes as input a private key $sk_{I_{key}}$ and a normal ciphertext ct. It outputs a message m or a failure symbol $\perp$.

$\mathsf{GenTK}_{out}(sk_{I_{key}})$. The transformation key generation algorithm takes as input a private key $sk_{I_{key}}$. It outputs a transformation key $tk_{I_{key}}$ and the corresponding retrieving key $rk_{I_{key}}$.

$\mathsf{Transform}_{out}(ct, tk_{I_{key}})$. The ciphertext transformation algorithm takes as input a ciphertext ct and a transformation key $tk_{I_{key}}$. It outputs a partially-decrypted ciphertext pct.

$\mathsf{Decrypt}_{out}(ct, pct, rk_{I_{key}})$. The outsourcing decryption algorithm takes as input a normal ciphertext ct, a partially-decrypted ciphertext pct and a retrieving key $rk_{I_{key}}$. It outputs a message m or a failure symbol $\perp$.

Let $(params, msk) \leftarrow \mathsf{Setup}(1^\lambda, U), sk_{I_{key}} \leftarrow \mathsf{KeyGen}(msk, I_{key}), ct \leftarrow \mathsf{Encryption}(m, I_{enc}), (rk_{I_{enc}}, tk_{I_{enc}}) \leftarrow \mathsf{GenTK}_{out}(sk_{I_{key}}), pct \leftarrow \mathsf{Transform}_{out}(ct, tk_{I_{key}})$. The correctness of an ABE scheme with outsourced decryption requires that, if $R(I_{enc}, I_{key}) = 1$, then both $\mathsf{Decrypt}(sk_{I_{key}}, ct)$ and $\mathsf{Decrypt}_{out}(ct, pct, rk_{I_{key}})$ output m. Otherwise, both $\mathsf{Decrypt}(sk_{I_{key}}, ct)$ and $\mathsf{Decrypt}_{out}(ct, pct, rk_{I_{key}})$ output a failure symbol $\perp$.

We now consider the privacy requirement. That is, any adversary including the third party should not learn anything about the encrypted message. In ABE with outsourced decryption, we consider Replayable CCA (RCCA) security [5], [8], [36]. Informally, it allows modifications to a given valid ciphertext, but the adversary cannot change the underlying message in a meaningful way. The RCCA security is formally described by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In the RCCA-security game, $\mathcal{A}$ is allowed to issue private key queries, transformation key queries, decryption queries and outsourcing decryption queries. Due to the space limitation, we omit the formal definition of the RCCA-security game, which can be referred to [8].

Definition 1 (IND-RCCA, [8]). *An ABE scheme with outsourced decryption is said to be IND-RCCA secure, if all polynomial time adversaries have at most a negligible advantage in the RCCA-security game.*

We also define two weak security notions for ABE with outsourced decryption: CPA security and selective security. An ABE scheme with outsourced decryption is said to be *secure against chosen-plaintext attacks* (CPA-secure) if both the decryption oracle and the outsourcing decryption oracle are removed in the query phases. An ABE scheme with outsourced decryption is said to be *selectively secure* if we add an Init phase before Setup, where the adversary submits the challenge value I^*_{enc} to the challenger.

We then consider the verifiability requirement of ABE with outsourced decryption. Loosely speaking, an ABE scheme with outsourced decryption is an ABE scheme with *verifiable* outsourced decryption (or, is said to be verifiable), if the correctness of the transformation done by a proxy can be verified. Similarly, the verifiability requirement is also formally described by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. We omit the formal definition of the verifiability game, which also can be referred to [8].

Definition 2 (Verifiability of Outsourced Decryption, [8]). *An ABE scheme with outsourced decryption is an ABE scheme with* verifiable *outsourced decryption (or, is said to be verifiable) if all polynomial time adversaries have at most a negligible advantage in the verifiability game.*

2.5 Ciphertext Verifiability and Delegatability of ABE with Outsourced Decryption

We consider the generic construction of RCCA-secure ABE with verifiable outsourced decryption. In its security proof, since the underlying ABE with outsourced decryption is CPA-secure and cannot provide a decryption oracle, we need another method to answer decryption queries. Here we borrow the idea proposed by Yamada et al. [10] and need the underlying ABE scheme with outsourced decryption to have either ciphertext verifiability or delegatability.

Informally, say an ABE scheme with outsourced decryption has the ciphertext verifiability property if it is possible to verify whether a *normal* ciphertext will be recovered into the same plaintext under two different decryption keys with two specific attributes. The formal definition is as follows.

Definition 3 (Ciphertext verifiability). *An ABE scheme with outsourced decryption is said to have ciphertext verifiability if there exists a polynomial time algorithm* Verify *that takes as input the (omitted) public parameters $params$, a normal ciphertext ct and two distinct values I_{key}, I'_{key}, and outputs $b \in \{0, 1, \perp\}$ according to the following properties. Let I_{enc} be obtained from parsing the ciphertext ct and $(params, msk) \leftarrow \mathsf{Setup}(1^\lambda, U)$.*

If $R(I_{key}, I_{enc}) \wedge R(I'_{key}, I_{enc}) = 0$, then Verify *outputs $\perp$.*

If $R(I_{key}, I_{enc}) \wedge R(I'_{key}, I_{enc}) = 1$, then:

*(*Soundness*) For all ct (which might be invalid) in the normal ciphertext space,*

$$\Pr\left[\begin{array}{r}\mathsf{Verify}(ct, I_{key}, I'_{key}) = 1 \\ \mathsf{Decrypt}(sk_{I_{key}}, ct) : sk_{I_{key}} \leftarrow \mathsf{KeyGen}(msk, I_{key})] \\ = \mathsf{Decrypt}(sk_{I'_{key}}, ct) sk_{I'_{key}} \leftarrow \mathsf{KeyGen}(msk, I'_{key})]\end{array}\right] = 1.$$

*(*Completeness*) For all m in the message space,*

$$\Pr[\mathsf{Verify}(ct, I_{key}, I'_{key}) = 1 : ct \leftarrow \mathsf{Encrypt}(m, I_{enc})] = 1.$$

Remarks. Definitions 2 and 3 define two different verifiability capabilities of ABE systems with outsourced decryption respectively. The former depicts the capability to verify whether the outsourced decryption done by a proxy is done correctly, while the latter depicts the capability to verify whether a normal ciphertext will be recovered into the same plaintext under two different decryption keys, which should satisfy soundness and completeness defined above.

Informally, the delegatability property of ABE with outsourced decryption is the capability to use a key for I_{key} to derive another key for I'_{key}, which is possible if I'_{key} is inferior than I_{key} when considering a well-defined partial order relation. Here we abuse notation $\supseteq$ denoting the partial order relation for both attribute set and access policy.

Definition 4 (Delegatability). *An ABE scheme with outsourced decryption is said to have delegatability if there exists a polynomial time algorithm* Delegate *such that the output of* $\mathsf{Delegate}(params, sk_{I_{key}}, I_{key}, I'_{key})$ *and of* $\mathsf{KeyGen}(msk, I'_{key})$ *have the same probability distribution for all $sk_{I_{key}}$ output by* $\mathsf{KeyGen}(msk, I_{key})$ *and for all I_{key}, I'_{key} such that $I_{key} \supseteq I'_{key}$.*

3 Generic Constructions of ABE with Verifiable Outsourced Decryption

In this section, we shall give *generic* constructions of CPA-secure and RCCA-secure ABE systems with *verifiable* outsourced decryption from CPA-secure ABE system with

outsourced decryption respectively. Note that our constructions can be also applied to selectively CPA-secure ABE systems with outsourced decryption. And then, the resulting ABE systems with verifiable outsourced decryption are only selectively CPA-secure/RCCA-secure as well.

Unlike the technique of separately encrypting an extra random message and then using this random message to commit to the true message in [8], the method adopted in our CPA-secure construction is encrypting a message and a random value *together* and then committing to the message by using the random value. Our method brings significant benefits. First, our construction is generic, and it has more compact ciphertext and less computational costs. Second, such a generic CPA-secure construction can be transformed into a generic RCCA-secure construction more naturally.

3.1 Generic CPA-Secure Construction

Now, we give the generic construction of CPA-secure ABE with *verifiable* outsourced decryption from CPA-secure ABE with outsourced decryption. As we have said, in this construction, we employ a commitment scheme to verify the correctness of the outsourced decryption.

Let $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}', \mathsf{GenTK}'_{out}, \mathsf{Transform}'_{out}, \mathsf{Decrypt}'_{out})$ be an ABE scheme with outsourced decryption and let $\mathcal{CS} = (\mathsf{InitCom}, \mathsf{Commit}, \mathsf{Reveal})$ be a (non-interactive) commitment scheme. We construct an ABE scheme with *verifiable* outsourced decryption $\Pi_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{GenTK}_{out}, \mathsf{Transform}_{out}, \mathsf{Decrypt}_{out})$ as follows:

$\mathsf{Setup}(1^\lambda, U)$. It runs $\mathsf{InitCom}(1^\lambda)$ to generate cp and runs $\mathsf{Setup}'(1^\lambda, U)$ to generate $(params, msk)$. The public parameters are $(params, cp)$ and the master secret key is msk.

$\mathsf{KeyGen}(msk, I_{key})$. It runs $\mathsf{KeyGen}'(msk, I_{key})$ to generate $sk_{I_{key}}$. The private key is $sk_{I_{key}}$.

$\mathsf{Encrypt}(m, I_{enc})$. To encrypt a message m under the value I_{enc}, it first chooses a random value r and then encrypts the "message" $m\|r$. That is, it runs $ct' \leftarrow \mathsf{Encrypt}'(m\|r, I_{enc})$. Then, it commits to m using the value r. That is, it runs $cm \leftarrow \mathsf{Commit}(m, r)$. The final ciphertext is $\langle ct', cm\rangle$.

$\mathsf{Decrypt}(sk_{I_{key}}, ct)$. To decrypt a ciphertext ct, it first parses ct as $\langle ct', cm\rangle$ and then runs $\mathsf{Decrypt}'(sk_{I_{key}}, ct')$. This yields a "message" $m\|r$. If $\mathsf{Reveal}(m, r, cm) = 1$ holds, it outputs m. Otherwise it outputs $\perp$.

$\mathsf{GenTK}_{out}(sk_{I_{key}})$. It runs $\mathsf{GenTK}'_{out}(sk_{I_{key}})$ to generate $(tk_{I_{key}}, rk_{I_{key}})$. The transformation key is $tk_{I_{key}}$ and the corresponding retrieving key is $rk_{I_{key}}$.

$\mathsf{Transform}_{out}(ct, tk_{I_{key}})$. It parses the normal ciphertext ct as $\langle ct', cm\rangle$ and runs $\mathsf{Transform}'_{out}(ct', tk_{I_{key}})$ to generate pct. The partially-decrypted ciphertext is pct.

$\mathsf{Decrypt}_{out}(ct, pct, rk_{I_{key}})$. It parses the normal ciphertext ct as $\langle ct', cm\rangle$ and runs $\mathsf{Decrypt}'_{out}(ct', pct, rk_{I_{key}})$. This yields a "message" $m\|r$. If $\mathsf{Reveal}(m, r, cm) = 1$ holds, it outputs m. Otherwise, it outputs $\perp$.

In the partially-decrypted ciphertext, we ignore the components that appears in the normal ciphertext. It is very obvious that this new ABE scheme with verifiable outsourced decryption Π_1 satisfies correctness.

3.2 Security Proofs of CPA-Secure Construction

In this subsection, we give the security proofs for the construction described above. First, we give the CPA-security proof for the above construction as follows:

Theorem 1. *Assume that the underlying ABE scheme with outsourced decryption is (selectively) CPA-secure and the underlying commitment scheme is computationally hiding, then the above construction of ABE scheme with verifiable outsourced decryption is (selectively) CPA-secure.*

Proof. We assume that the equal-length challenge messages which the adversary submits in the challenge phase have length l_1. We also assume that the random value r used in algorithm $\mathsf{Encrypt}$ has fixed length l_2. The CPA security of the above scheme can be proved by defining the following hybrid games:

- Game_0: The original CPA-security game of ABE with verifiable outsourced decryption. Note that when the adversary submits two equal-length challenge messages m_0, m_1 and a value I^*_{enc}, the challenger flips a random coin $b \in \{0, 1\}$, selects a random value r^* and then computes the challenge ciphertext components ct'^* and cm^* by running $ct'^* \leftarrow \mathsf{Encrypt}'(m_b\|r^*, I^*_{enc})$ and $cm^* \leftarrow \mathsf{Commit}(m_b, r^*)$, respectively.
- Game_1: Same as Game_0 except that when $\mathcal{A}$ submits two equal-length challenge messages, the challenger sets the component ct'^* of the challenge ciphertext by running $ct'^* \leftarrow \mathsf{Encrypt}'(0^{l_1+l_2}, I^*_{enc})$.
- Game_2: Same as Game_1 except that when $\mathcal{A}$ submits two equal-length challenge messages, the challenger sets the component cm^* of the challenge ciphertext by running $cm^* \leftarrow \mathsf{Commit}(0^{l_1}, r^*)$.

We prove that based on the security of different primitives, the advantage of any PPT adversary in each game must be negligibly close to that in the previous game in the following two lemmas. In the last game, since the challenge ciphertext is independent of the challenge messages chosen by the adversary, the adversary has no advantage. Therefore, we conclude that the adversary in the original IND-CPA security game (Game_0) has a negligible advantage. This completes the proof of Theorem 1. □

Lemma 1. *If the underlying ABE scheme with outsourced decryption is CPA-secure, then no PPT adversary can attain a non-negligible difference in advantage between* Game_0 *and* Game_1.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that exhibits a non-negligible difference in advantage between Game_0 and Game_1, then we can build a simulator $\mathcal{B}$ that attacks the underlying ABE scheme with outsourced decryption in the CPA-security game with a non-negligible advantage.

Let $\mathcal{C}$ be the corresponding challenger of $\mathcal{B}$ in the CPA security game of the underlying ABE scheme with outsourced decryption. $\mathcal{B}$ runs $\mathcal{A}$ to execute the following steps.

Setup. The challenger $\mathcal{C}$ runs $(params, msk) \leftarrow \mathsf{Setup}'(1^\lambda, U)$. The simulator $\mathcal{B}$ is given only the public parameters $params$ and runs $cp \leftarrow \mathsf{InitCom}(1^\lambda)$. $\mathcal{B}$ then sends the public parameters $(params, cp)$ to $\mathcal{A}$.

Query Phase I. When the adversary $\mathcal{A}$ queries the private key (resp. the transformation key) on I_{key}, the simulator $\mathcal{B}$ passes I_{key} on to its own private key generation oracle (resp. transformation key generation oracle) and forwards the returned value to $\mathcal{A}$.

Challenge. The adversary $\mathcal{A}$ submits two equal-length challenge messages m_0, m_1 of length l_1. It also submits a value I^*_{enc} such that $R(I_{key}, I^*_{enc}) = 0$ for all I_{key} submitted to the private key generation oracle. The simulator $\mathcal{B}$ selects a random value r^*, flips a random coin $b \in \{0,1\}$ and computes $cm^* \leftarrow \mathsf{Commit}(m_b, r^*)$. It then sends $(M_0, M_1) = (m_b \| r^*, 0^{l_1+l_2})$ and I^*_{enc} to its encryption oracle. The challenger $\mathcal{C}$ flips a random coin $\beta \in \{0,1\}$ and runs $ct'^* \leftarrow \mathsf{Encrypt}'(M_\beta, I^*_{enc})$. Lastly, the simulator $\mathcal{B}$ is given ct'^* and sends $\mathcal{A}$ the challenge ciphertext $\langle ct'^*, cm^* \rangle$.

Query Phase II. This phase is identical to Query Phase I. The only constraint is that $\mathcal{A}$ cannot request a private key query on I_{key} which satisfies $R(I_{key}, I^*_{enc}) = 1$.

Guess. The adversary $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0. Otherwise, it outputs 1.

It is a remarkable fact that, if the encryption query of $\mathcal{B}$ is answered with an encryption of $m_b \| r^*$, then the view of $\mathcal{A}$ is exactly as in Game_0, otherwise the view of $\mathcal{A}$ is exactly as in Game_1. Thus, if $\mathcal{A}$ can distinguish Game_0 and Game_1 with a non-negligible advantage, we can build an algorithm $\mathcal{B}$ that attacks the underlying CPA-secure ABE scheme with outsourced decryption with a non-negligible advantage. This completes the proof of Lemma 1. □

Lemma 2. *If the underlying commitment scheme is computationally hiding, then no PPT adversary can attain a non-negligible difference in advantage between* Game_1 *and* Game_2.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that exhibits a non-negligible difference in advantage between Game_1 and Game_2, then we can build a simulator $\mathcal{B}$ that attacks the hiding property of the underlying commitment scheme with a non-negligible advantage.

Let $\mathcal{C}$ be the corresponding challenger of $\mathcal{B}$ in the hiding property game of the underlying commitment scheme. $\mathcal{B}$ runs $\mathcal{A}$ to execute the following steps.

Setup. The challenger $\mathcal{C}$ runs $cp \leftarrow \mathsf{InitCom}(1^\lambda)$. Given cp, $\mathcal{B}$ runs $(params, msk) \leftarrow \mathsf{Setup}'(1^\lambda, U)$ and then sends the public parameters $(params, cp)$ to $\mathcal{A}$.

Query Phase I. Since the simulator $\mathcal{B}$ has the master secret key msk, private key queries and transformation key queries can be answered in the natural way.

Challenge. The adversary $\mathcal{A}$ submits two equal-length challenge messages m_0, m_1 of length l_1. It also submits a value I^*_{enc} such that $R(I_{key}, I^*_{enc}) = 0$ for all I_{key} submitted to the private key generation oracle. The simulator $\mathcal{B}$ first computes $ct'^* \leftarrow \mathsf{Encrypt}'(0^{l_1+l_2}, I^*_{enc})$. $\mathcal{B}$ then flips a random coin $b \in \{0,1\}$ and sends $(M_0, M_1) = (m_b, 0^{l_1})$ to the challenger $\mathcal{C}$. $\mathcal{C}$ flips a random coin $\beta \in \{0,1\}$ and computes $cm^* \leftarrow \mathsf{Commit}(M_\beta, r^*)$, where r^* is chosen from some proper domain. After receiving the commitment cm^*, $\mathcal{B}$ sends $\mathcal{A}$ the challenge ciphertext $\langle ct'^*, cm^* \rangle$.

Query Phase II. Further queries can be issued by $\mathcal{A}$ and then can be answered by $\mathcal{B}$ in the natural way. The only constraint is that $\mathcal{A}$ cannot request a private key query on I_{key} which satisfies $R(I_{key}, I^*_{enc}) = 1$.

Guess. The adversary $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0. Otherwise, it outputs 1.

Obviously, if $\beta = 0$ then the view of $\mathcal{A}$ is exactly as in Game_1, while if $\beta = 1$ then the view of $\mathcal{A}$ is exactly as in Game_2. Thus, if $\mathcal{A}$ can distinguish Game_1 and Game_2 with a non-negligible advantage, we can build an algorithm $\mathcal{B}$ that breaks the hiding property of the underlying commitment scheme with a non-negligible advantage. Thus, this completes the proof of Lemma 2. □

Now, we give the proof of the verifiability of outsourced decryption for the CPA-secure construction Π_1 as follows:

Theorem 2. *If the underlying commitment scheme is computationally binding, then the above construction of ABE scheme with verifiable outsourced decryption is verifiable.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ that attacks the above ABE scheme with verifiable outsourced decryption in the verifiability game with a non-negligible advantage, than we can build a simulator $\mathcal{B}$ which breaks the binding property of the underlying commitment scheme with a non-negligible advantage. $\mathcal{B}$ runs $\mathcal{A}$ to execute the following steps. Given the commitment parameters cp, $\mathcal{B}$ runs $(params, msk) \leftarrow \mathsf{Setup}'(1^\lambda)$ and sends the public parameters $(params, cp)$ to $\mathcal{A}$. Since the simulator $\mathcal{B}$ has the master secret key msk, any query issued by $\mathcal{A}$ can be answered in the natural way. When $\mathcal{A}$ submits a message m^* and a value I^*_{enc}, $\mathcal{B}$ returns to $\mathcal{A}$ the challenge ciphertext $ct^* = \langle ct'^*, cm^* \rangle$ in the expected way. Further queries can be issued by $\mathcal{A}$ and then can be answered by $\mathcal{B}$ as above. Lastly, $\mathcal{A}$ outputs a value I^*_{key} and a partially-decrypted ciphertext pct^*. We make a reasonable assumption that $\mathcal{A}$ has issued a transformation key query on I^*_{key} (Notice that if not, $\mathcal{B}$ itself can issue such a transformation key query on I^*_{key}). Then $\mathcal{B}$ knows the quadruple $(I^*_{key}, sk_{I^*_{key}}, tk_{I^*_{key}}, rk_{I^*_{key}})$. $\mathcal{A}$ wins the game if $\mathsf{Decrypt}_{out}(ct^*, pct^*, rk_{I^*_{key}}) \notin \{m^*, \perp\}$.

If the adversary $\mathcal{A}$ wins in the above game, then we have that $m' \| r' \leftarrow \mathsf{Decrypt}'_{out}(ct'^*, pct^*, rk_{I^*_{key}})$ and $\mathsf{Reveal}(m', r', cm^*) = 1$, where $m' \notin \{m^*, \perp\}$. Thus, the simulator $\mathcal{B}$ reveals two different messages m' and m^* from the commitment cm^*. It exactly violates the binding property of the underlying commitment scheme. This completes the proof of Theorem 2. □

3.3 Moving on to Generic RCCA-Secure Construction

In this subsection, we give the generic construction of RCCA-secure ABE with *verifiable* outsourced decryption from CPA-secure ABE with outsourced decryption.

TABLE 1
How to Setup I'_{key}, I'_{enc} and sk_t in Each Case

	CP-ABE w/ Out. Dec.	KP-ABE w/ Out. Dec.
I'_{key}	I_{key}	I_{key}
I'_{enc}	$I_{enc} \vee A$	$I_{enc} \cup S_{com}$
sk_t	$b \leftarrow \mathsf{Verify}(ct', I_{key}, S_{com})$ if $b = 1$, $sk_t = sk_{I_{key}}$ else $sk_t = \perp$	$b \leftarrow \mathsf{Verify}(ct', I_{key}, A)$ if $b = 1$, $sk_t = sk_{I_{key}}$ else $sk_t = \perp$
I'_{key}	$I_{key} \cup W$	I_{key}
I'_{enc}	$I_{enc} \wedge A$	$I_{enc} \cup S_{com}$
sk_t	$X = I_{key} \cup W$ $Y = I_{key} \cup S_{com}$ $sk_t \leftarrow \mathsf{Delegate}(B, X, Y)$	$X = I_{key}$ $Y = I_{key} \wedge A$ $sk_t \leftarrow \mathsf{Delegate}(B, X, Y)$

Note. Here $A = (\wedge_{P \in S_{com}} P)$ and $B = sk_{I_{key}}$. Above is the case that the underlying CPA-secure ABE scheme with outsourced decryption has ciphertext verifiability, while below is the case that the underlying CPA-secure ABE scheme with outsourced decryption has delegatability.

Suppose that we shall construct an ABE scheme with verifiable outsourced decryption to work with a universe U. A set W of dummy attributes, which is disjointed from U, is used in the construction. The underlying CPA-secure ABE scheme with outsourced decryption is then required to work with universe $U \cup W$. A dummy attribute set $S_{com} \subset W$ is associated to an encapsulation string com of an encapsulation scheme. We assume that for all com, $com \in \{0,1\}^l$.

As in [10], the sets W and S_{com} are defined as follows. If the underlying ABE with outsourced scheme is a small universe scheme, we set $W = \{P_{i,0}, P_{i,1}\}_{i \in [l]}$. We then generate dummy attribute set $S_{com} \subset W$ by letting $S_{com} = \{P_{i,com_i}\}_{i \in [l]}$, where com_i is the ith bit of com. If the underlying ABE with outsourced scheme is a large universe scheme, we set $W = \{0,1\}^l$. We then generate dummy attribute set $S_{com} \subset W$ by simply letting $S_{com} = \{com\}$.

Let $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}', \mathsf{GenTK}'_{out}, \mathsf{Transform}'_{out}, \mathsf{Decrypt}'_{out})$ be an ABE scheme with outsourced decryption which has ciphertext verifiability or delegatability, let $\mathcal{ES} = (\mathsf{Init}, \mathsf{Encap}, \mathsf{Decap})$ be an encapsulation scheme, let $\mathcal{MA} = (\mathsf{Mac}, \mathsf{Ver})$ be a message authentication code, and let $\mathcal{CS} = (\mathsf{InitCom}, \mathsf{Commit}, \mathsf{Reveal})$ be a (non-interactive) commitment scheme. We construct an ABE scheme with *verifiable* outsourced decryption $\Pi_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{GenTK}_{out}, \mathsf{Transform}_{out}, \mathsf{Decrypt}_{out})$ as follows:

$\mathsf{Setup}(\lambda, U)$. It runs $\mathsf{InitCom}(1^\lambda)$ to generate cp, runs $\mathsf{Init}(\lambda)$ to generate pub and runs $\mathsf{Setup}'(\lambda, U \cup W)$ to generate $(params, msk)$. The master secret key is msk and the public parameters are $(params, pub, cp)$.

$\mathsf{KeyGen}(msk, I_{key})$. It sets I'_{key} according to Table 1 and runs $\mathsf{KeyGen}'(msk, I'_{key})$ to generate $sk_{I_{key}}$. The private key is $sk_{I_{key}}$.

$\mathsf{Encrypt}(m, I_{enc})$. To encrypt a message m, it first sets I'_{enc} according to Table 1. It then runs $\mathsf{Encap}(pub)$ to generate (r, com, dec) and encrypts the "message" $m \| dec$ by running $ct' \leftarrow \mathsf{Encrypt}'(m \| dec, I'_{enc})$. Next, it splits r into two equal-length strings r_1 and r_2, denoted by $r = r_1 \| r_2$. Lastly, it runs $\mathsf{Mac}_{r_1}(ct')$ to obtain an authentication code tag and runs $\mathsf{Commit}(m, r_2)$ to generate a commitment cm. The final ciphertext is $ct = \langle com, ct', tag, cm \rangle$.

$\mathsf{Decrypt}(sk_{I_{key}}, ct)$. To decrypt a normal ciphertext $ct = \langle com, ct', tag, cm \rangle$, it first generates an intermediate key sk_t according to Table 1. If $sk_t = \perp$, it returns $\perp$. Otherwise, it runs algorithm $\mathsf{Decrypt}'(sk_t, ct')$. If the result is $\perp$, it outputs $\perp$ as well. Otherwise, this yields a "message" $m \| dec$. It then runs $r \leftarrow \mathsf{Decap}(com, dec)$ and parses r as $r = r_1 \| r_2$. If $r \neq \perp$, $\mathsf{Ver}_{r_1}(ct', tag) = 1$ and $\mathsf{Reveal}(m, r_2, cm) = 1$, it outputs m. Otherwise it outputs $\perp$.

$\mathsf{GenTK}_{out}(sk_{I_{key}})$. It runs algorithm $\mathsf{GenTK}'_{out}(sk_{I_{key}})$ to generate $(tk_{I_{key}}, rk_{I_{key}})$. The transformation key is $tk_{I_{key}}$ and the corresponding retrieving key is $rk_{I_{key}}$.

$\mathsf{Transform}_{out}(ct, tk_{I_{key}})$. It first parses the normal ciphertext ct as $\langle com, ct', tag, cm \rangle$ and then runs algorithm $\mathsf{Transform}'_{out}(ct', tk_{I_{key}})$ to generate pct. The partially-decrypted ciphertext is pct.

$\mathsf{Decrypt}_{out}(ct, pct, rk_{I_{key}})$. It first parses the normal ciphertext ct as $\langle com, ct', tag, cm \rangle$ and then runs algorithm $\mathsf{Decrypt}'_{out}(ct', pct, rk_{I_{key}})$. If the result is $\perp$, it outputs $\perp$ as well. Otherwise, this yields a "message" $m \| dec$. It then runs $r \leftarrow \mathsf{Decap}(com, dec)$ and parses r as $r = r_1 \| r_2$. If $r \neq \perp$, $\mathsf{Ver}_{r_1}(ct', tag) = 1$ and $\mathsf{Reveal}(m, r_2, cm) = 1$, it outputs m. Otherwise it outputs $\perp$.

In the partially-decrypted ciphertext, we ignore the components that appears in the normal ciphertext. It is very obvious that the proposed ABE scheme with verifiable outsourced decryption Π_2 satisfies correctness.

Now, we give the security proofs for the construction described above. First, we give the RCCA-security proof for the above construction as follows:

Theorem 3. *Assume that the underlying ABE scheme with outsourced decryption, which has the property of ciphertext verifiability or delegatability, is (selectively) CPA-secure, the underlying encapsulation scheme is computationally hiding and computationally binding, the underlying message authentication code is a strong one-time message authentication code, and the underlying commitment scheme is computationally hiding, then the above construction of ABE scheme with verifiable outsourced decryption is (selectively) RCCA-secure.*

The formal proof will be given in Appendix.

We then show that the above RCCA-secure construction Π_2 is verifiable as follows:

Theorem 4. *If the underlying commitment scheme is computationally binding, then the above construction of ABE scheme with verifiable outsourced decryption is verifiable.*

The proof is extremely similar to the proof of Theorem 2. Since the simulator has the master secret key, any query can be answered in the natural way. Finally, the simulator is able to reveal two different messages from a commitment, which exactly violates the binding property of the underlying commitment scheme. We omit the formal proof due to the space limitation.

REMARKS. As we know, an ABE scheme with outsourced decryption just adds a transformation procedure to a

traditional ABE scheme in essence. So in our RCCA-secure construction, if we remove the component cm in the ciphertext, then the algorithms Setup, KeyGen, Encrypt and Decrypt constitute a CCA-secure ABE scheme. Hence, we obtain a generic transformation from (selectively) CPA-secure ABE scheme to (selectively) CCA-secure ABE scheme. We omit the proof, since it is quite similar to the proof of Theorem 3.

4 Instantiation and Performance

In this section, we instantiate our CPA-secure construction in the standard model and then evaluate its performance through implementations based on Charm. We emphasize again that a RCCA-secure scheme in the *standard* model can be obtained by instantiating our RCCA-secure construction.

4.1 Selectively CPA-Secure Instantiation

Based on Green et al.'s (small-universe and backwards-compatible variant[1] of) selectively CPA-secure CP-ABE scheme with outsourced decryption, we construct a selectively CPA-secure CP-ABE scheme with *verifiable* outsourced decryption. In this construction, we use the hybrid encryption method and employ Pedersen Commitment [11]. Note that Pedersen Commitment is computationally binding and perfectly hiding. We describe our instantiation as follows:

Setup$(1^\lambda, U)$. On input a security parameter λ and a description of (small) attribute universe U, the system setup algorithm first runs $\mathcal{G}(\lambda)$ to obtain a description of a bilinear group $(p, \mathbb{G}, \mathbb{G}_\mathrm{T}, e)$, where $\mathbb{G}$ and $\mathbb{G}_\mathrm{T}$ are (multiplicative) cyclic groups of prime order p. It then chooses a generator g of group $\mathbb{G}$, picks exponents α, a and $\{s_i\}_{i\in U}$ from $\mathbb{Z}_p$ uniformly at random, and selects elements $g_1, g_2 \in \mathbb{G}$ uniformly at random. Let $G : \mathbb{G}_\mathrm{T} \rightarrow \{0,1\}^*$ be a pseudorandom generator with sufficiently-long output length (e.g., constructed by first hashing elements of $\mathbb{G}_\mathrm{T}$ and then using a suitable modification of a block cipher). Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be a collision-resistant cryptographic hash function. The public parameters are published as $params = (p, \mathbb{G}, \mathbb{G}_\mathrm{T}, e, g, g_1, g_2, g^a, e(g,g)^\alpha, \{T_i = g^{s_i}\}_{i\in U}, G, H)$. The master secret key is kept as $msk = \alpha$.

KeyGen$(msk, \mathcal{S})$. On input the master security key msk and a set of attributes $\mathcal{S}$, the key generation algorithm picks an exponent $t \in \mathbb{Z}_p$ uniformly at random, and outputs the private key as $sk_\mathcal{S} = (\mathcal{S}, K = g^\alpha g^{at}, K_0 = g^t, \{K_i = T_i^t\}_{i\in\mathcal{S}})$.

Encryption$(m, \mathbb{A})$. On input a message m and an LSSS access structure $\mathbb{A} = (\mathbf{A}, \rho)$, where $\mathbf{A}$ is an $l \times n$ matrix and ρ is a map from each row A_i of $\mathbf{A}$ to an attribute $\rho(i)$, the encryption algorithm first chooses random $r \in \{0,1\}^*$, and picks random exponents $\vec{v} = (s, v_2, \ldots, v_n) \in \mathbb{Z}_p^n$ and $r_i \in \mathbb{Z}_p$ for each row A_i of $\mathbf{A}$. Next, it computes $C = (m\|r) \oplus G(e(g,g)^{\alpha s})$, $C_0 = g^s$, $\{C_{1,i} = g^{aA_i\vec{v}} T_{\rho(i)}^{r_i}\}_{i\in[l]}$, $\{C_{2,i} = g^{r_i}\}_{i\in[l]}$ and $cm = g_1^{H(m)} g_2^{H(r)}$. Lastly, it outputs the ciphertext $ct = (\mathbb{A}, C, C_0, \{C_{1,i}\}_{i\in[l]}, \{C_{2,i}\}_{i\in[l]}, cm)$.

1. Based on Waters' small-universe CP-ABE scheme [9] and Green et al.'s key blinding technique [5], such a variant can be easily constructed.

Decrypt$(sk_\mathcal{S}, ct)$. The decryption algorithm takes as input a private key $sk_\mathcal{S} = (\mathcal{S}, K, K_0, \{K_i\}_{i\in\mathcal{S}})$ and a ciphertext $ct = (\mathbb{A}, C, C_0, \{C_{1,i}\}_{i\in[l]}, \{C_{2,i}\}_{i\in[l]}, cm)$. If $\mathcal{S}$ does not satisfy the access structure $\mathbb{A}$, it outputs $\perp$. Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$ and let $I \subset [l]$ be defined as $I = \{i : \rho(i) \in \mathcal{S}\}$. It then computes constants $\omega_i \in \mathbb{Z}_p$ for $i \in I$ such that $\sum_{i\in I} \omega_i \mathbf{A}_i = (1, 0, \ldots, 0)$. Lastly, it computes

$$Z = \frac{e(C_0, K)}{\prod_{i\in I}(e(C_{1,i}, K_0) \cdot e(C_{2,i}, K_{\rho(i)}))^{\omega_i}} = e(g,g)^{\alpha s}$$

and $(m\|r) = C \oplus G(Z)$. If $g_1^{H(m)} g_2^{H(r)} = cm$ holds, it outputs m. Otherwise, it outputs $\perp$.

GenTK$_{out}(sk_\mathcal{S})$. On input a private key $sk_\mathcal{S} = (\mathcal{S}, K, K_0, \{K_i\}_{i\in\mathcal{S}})$, the transformation key generation algorithm first chooses a random exponent $z \in \mathbb{Z}_p$. Note that z has multiplicative inverse with overwhelming probability. It then sets the transformation key as $tk_\mathcal{S} = (\mathcal{S}, K' = K^{1/z}, K_0' = K_0^{1/z}, \{K_i' = K_i^{1/z}\}_{i\in\mathcal{S}})$ and the corresponding retrieving key as $rk_\mathcal{S} = z$.

Transform$_{out}(ct, tk_\mathcal{S})$. The ciphertext transformation algorithm takes as input a normal ciphertext $ct = (\mathbb{A}, C, C_0, \{C_{1,i}\}_{i\in[l]}, \{C_{2,i}\}_{i\in[l]}, cm)$ and a transformation key $tk_\mathcal{S} = (\mathcal{S}, K', K_0', \{K_i'\}_{i\in\mathcal{S}})$ for $\mathcal{S}$. It computes the partially-decrypted ciphertext pct as

$$pct = \frac{e(C_0, K')}{\prod_{i\in I}(e(C_{1,i}, K_0') \cdot e(C_{2,i}, K_{\rho(i)}'))^{\omega_i}} = e(g,g)^{\alpha s/z}.$$

Here we ignore the components $(\mathbb{A}, C, cm)$ which appears in the normal ciphertext.

Decrypt$_{out}(ct, pct, rk_\mathcal{S})$. On input a normal ciphertext $ct = (\mathbb{A}, C, C_0, \{C_{1,i}\}_{i\in[l]}, \{C_{2,i}\}_{i\in[l]}, cm)$, a partially-decrypted ciphertext $pct = e(g,g)^{\alpha s/z}$ and a retrieving key $rk_\mathcal{S} = z$ for an attributes set $\mathcal{S}$, the outsourcing decryption algorithm computes $(m\|r) = C \oplus G(pct^z)$. If $g_1^{H(m)} g_2^{H(r)} = cm$ holds, it outputs m. Otherwise, it outputs $\perp$.

It is not difficult to see that this concrete scheme satisfies the correctness requirement. Regarding secuirty, the underlying CP-ABE scheme with outsourced decryption is obtained by applying Green et al.'s key blinding technique [5] to Waters' small-universe CP-ABE scheme [9]. Since Waters' CP-ABE scheme is selectively CPA-secure and Pedersen Commitment is computationally binding and perfectly hiding, this scheme is selectively CPA-secure in the standard model and verifiable according to Theorem 1 and Theorem 2.

4.2 Performance

We implemented both the above selectively CPA-secure instantiation and Lai et al.'s scheme [8] in software based on Charm [13] (version 0.41b). As in [5], we implemented these two schemes using a MNT224 elliptic curve, which implies the use of asymmetric groups equipped with a pairing e of the form $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\mathrm{T}$. In our implementations, we also used the hybrid encryption method as in [5]. We first implemented a key encapsulation mechanism (KEM) variant of Waters' CP-ABE scheme. A symmetric session key is then computed from this attribute-based KEM variant and all the

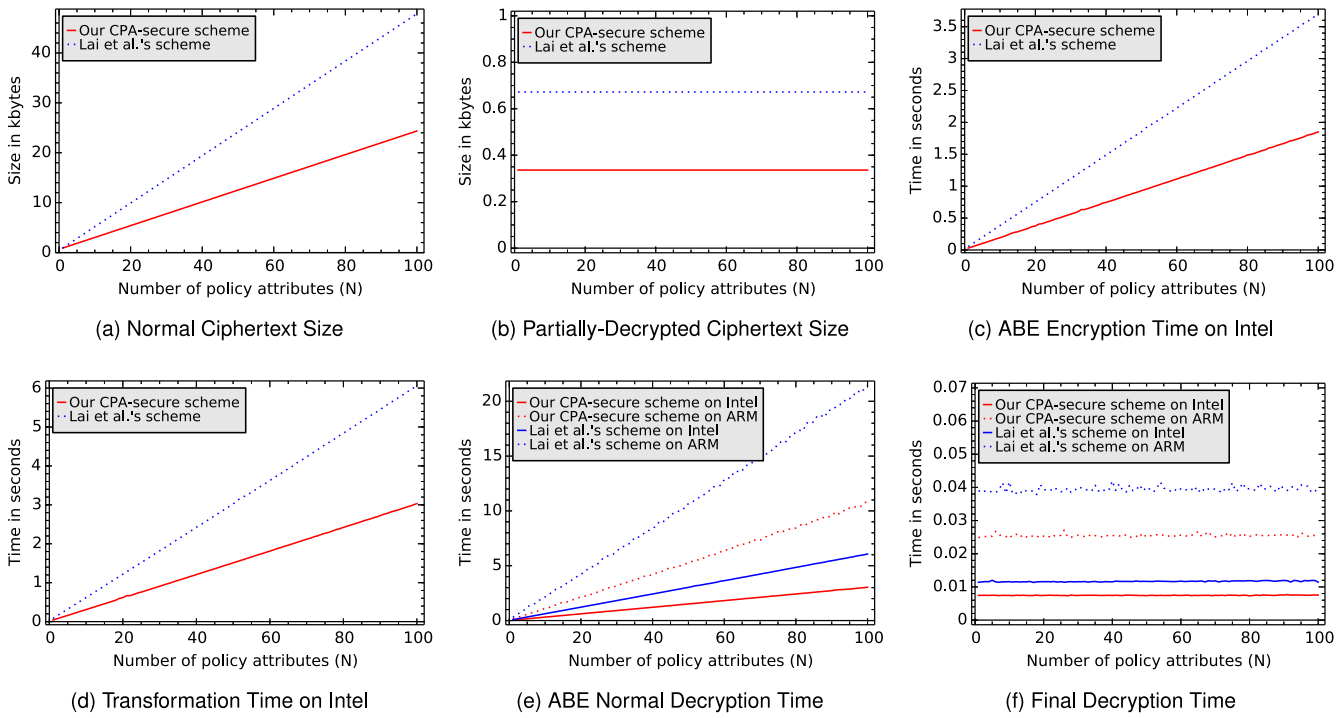


Fig. 2. Simulation results of our and Lai et al.'s selectively CPA-secure systems.

messages are encrypted under this session key by using a symmetric encryption scheme (In our experiments, we used AES encryption in CBC mode).

Our platform consists of a desktop computer and an Android mobile phone, which are modeled as the powerful proxy and the resource-limited data consumer respectively. Concretely, all our benchmarks were executed on these two dedicated hardware platforms, a desktop computer (equipped with a dual core Intel(R) CPU i3-2120@3.30 GHz and 2.0 GB RAM) running 32-bit Archlinux (Kernel version 3.17.3) and Python 3.4.2, and a Samsung Galaxy Note II smartphone (equipped with an ARM-based CPU Exynos-4412@1,638 MHz and 2.0 GB RAM) running Android OS 4.3, Scripting Layer for Android (SL4A) r5 and Python 3.

Overall, we adopted similar experimental methods as those described in [5] and [8]. We generated access policies in the form of $(A_1$ and $\ldots$ and $A_N)$, where each A_i is an attribute. By increasing values of N from 1 to 100, we obtained 100 different access polices. With each access policy, we then constructed a corresponding (normal) decryption key that exactly contains N attributes. For accuracy, we also repeated our experiments 100 times on the desktop computer and 30 times on the smartphone for each scheme, and then took the average values as the experimental results.

Fig. 2 illustrates the final experimental results. The normal ciphertext size in our scheme is much smaller than the one in Lai et al.'s scheme. The same situation happens to the partially-decrypted ciphertext size. In our scheme, to decrypt a normal ciphertext with policy consisting of 100 attributes, it takes almost 11 seconds on the smartphone and takes about 3 seconds on the desktop computer. The computational costs are much smaller than those in Lai et al.'s scheme. Another observation is that outsourcing decryption from the resource-limited smartphone to the powerful computer can significantly reduce the computational costs of the smartphone. As we see, with outsourced decryption, it only takes about 25 milliseconds to decrypt on the smartphone device in our scheme.

5 Conclusion

We have presented *generic* constructions of CPA-secure and RCCA-secure ABE with *verifiable* outsourced decryption from CPA-secure ABE with outsourced decryption, respectively. Note that the techniques involved in RCCA-secure construction can be applied in generally constructing CCA-secure ABE from CPA-secure ABE. We have instantiated the CPA-secure construction in the standard model. Another important fact is that a RCCA-secure ABE scheme with verifiable outsourced decryption in the *standard* model can be easily obtained from our generic RCCA-secure construction. We have then implemented our CPA-secure instantiation. The experimental results show that, compared with the existing selectively CPA-secure system, our instantiation has more compact ciphertext and less computational costs.

Appendix

Proof of Theorem 3

Proof. Here we shall give the proof of Theorem 3 for the case of CP-ABE with verifiable outsourced decryption (namely, $I_{key} = \mathcal{S}$ and $I_{enc} = \mathbb{A}$) and the underlying CP-ABE with outsourced decryption has the property of ciphertext verifiability. The theorem for other cases can be proved in a similar way. We assume that the equal-length challenge messages which the adversary submits in the challenge phase have length l_1. We also assume all the values dec output by algorithm Encap have fixed length l_3.

Let r^*, com^*, dec^* denote the values that are generated by $\mathsf{Encap}(pub)$ in computing the challenge ciphertext.

Since these values are generated independently of $\mathcal{A}$'s actions, we may assume these values are generated at the outset of the experiment. Assume that $r^* = r_1^* \| r_2^*$. Let Succ denote the event that the bit b' output by $\mathcal{A}$ is equal to the bit b used in constructing the challenge ciphertext. Let Valid denote the event that $\mathcal{A}$ at some point submits a valid normal ciphertext $\langle com^*, ct', tag, cm\rangle$ to its decryption or outsourcing decryption oracle. Say a ciphertext is "valid" if its decryption does not result in $\perp$. Let NoBind denote the event that $\mathcal{A}$ at some point submits a normal ciphertext $\langle com^*, ct', tag, cm\rangle$ to its decryption or outsourcing decryption oracle such that ct' decrypts to $m \| dec$ and $\mathsf{Decap}(pub, com^*, dec) = r' \notin \{r^*, \perp\}$. Let Forge denote the event that $\mathcal{A}$ at some point submits a normal ciphertext $\langle com^*, ct', tag, cm\rangle$ to its decryption or outsourcing decryption oracle such that $\mathsf{Ver}_{r_1^*}(ct', tag) = 1$. To prove this theorem, we define a sequence of games. Denote by $\Pr_i[\cdot]$ the probability of a particular event occurring in Game_i.

Game_0 is the original RCCA-security game of ABE with verifiable outsourced decryption. Note that when the adversary submits two equal-length challenge messages m_0, m_1 and an access structure $\mathbb{A}^*$, the challenger flips a random coin $b \in \{0,1\}$, generates another access structure $\mathbb{A}'^* = \mathbb{A}^* \vee (\wedge_{P \in S_{com^*}} P)$ according to Table 1, and then computes the challenge ciphertext components ct'^*, tag^* and cm^* by running $ct'^* \leftarrow \mathsf{Encrypt}'(m_b \| dec^*, \mathbb{A}'^*)$, $tag^* \leftarrow \mathsf{Mac}_{r_1^*}(ct'^*)$ and $cm^* \leftarrow \mathsf{Commit}(m_b, r_2^*)$, respectively.

Game_1 is same as Game_0 except that on input a normal ciphertext of the form $\langle com^*, ct', tag, cm\rangle$, the decryption oracle (or the outsourcing decryption oracle) simply outputs $\perp$. Obviously, Game_1 is identical to Game_0 until event Valid occurs and we also have $\Pr_0[\mathsf{Valid}] = \Pr_1[\mathsf{Valid}] \leq \Pr_1[\mathsf{NoBind}] + \Pr_1[\mathsf{Forge}]$. Note that this upper bound is relaxed since event Forge also contains the situation that algorithm Reveal of the underlying commitment outputs 0, which indicates that the ciphertext is not valid. Thus, $|\Pr_1[\mathsf{Succ}] - \Pr_0[\mathsf{Succ}]| \leq \Pr_1[\mathsf{NoBind}] + \Pr_1[\mathsf{Forge}]$.

We show that $\Pr_1[\mathsf{NoBind}]$ is negligible. Consider a simulator $\mathcal{B}$ which runs $\mathcal{A}$ to execute the following steps: given (pub, com^*, dec^*), $\mathcal{B}$ runs $(params, msk) \leftarrow \mathsf{Setup}'(1^\lambda, U \cup W)$, $cp \leftarrow \mathsf{InitCom}(1^\lambda)$ and sends the public parameters $(params, pub, cp)$ to $\mathcal{A}$. Since $\mathcal{B}$ has the master secret key msk, any query issued by $\mathcal{A}$ can be answered in the natural way. Specifically, on input a normal ciphertext of the form $\langle com^*, ct', tag, cm\rangle$, the decryption oracle (or the outsourcing decryption oracle) simply outputs $\perp$. When $\mathcal{A}$ submits two equal-length challenge messages m_0, m_1 and an access structure $\mathbb{A}^*$, $\mathcal{B}$ runs $r^* \leftarrow \mathsf{Decap}(pub, com^*, dec^*)$ and returns to $\mathcal{A}$ the challenge ciphertext in the expected way. Further queries with the restrictions defined in the security model can be issued by $\mathcal{A}$ and then can be answered by $\mathcal{B}$ in the natural way. Lastly, $\mathcal{B}$ just ignores $\mathcal{A}$'s guess. In this game, $\mathcal{B}$ can decrypt every query of the form $\langle com^*, ct', tag, cm\rangle$ since it has the master secret key. If event NoBind occurs, there exists at least a value dec' such that $\mathsf{Decap}(pub, com^*, dec') = r' \notin \{r^*, \perp\}$, which exactly violates the binding property of the underlying encapsulation scheme. Hence, $\Pr_1[\mathsf{NoBind}]$ must be negligible.

Game_2 is same as Game_1 except that the challenger sets the component ct'^* of the challenge ciphertext by running $ct'^* \leftarrow \mathsf{Encrypt}'(0^{l_1+l_3}, \mathbb{A}'^*)$. We show that $|\Pr_2[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]|$ is negligible. To see this, consider a simulator $\mathcal{B}$ attacking the underlying ABE scheme with outsourced decryption in the CPA-security game. Let $\mathcal{C}$ be the corresponding challenger of $\mathcal{B}$ in the CPA-security game of the ABE scheme. $\mathcal{B}$ runs $\mathcal{A}$ to execute the following steps.

Setup. $\mathcal{C}$ runs $(params, msk) \leftarrow \mathsf{Setup}'(1^\lambda, U \cup W)$. Given the public parameters $params$, $\mathcal{B}$ runs $pub \leftarrow \mathsf{Init}(\lambda)$, $cp \leftarrow \mathsf{InitCom}(1^\lambda)$ and $(r^*, com^*, dec^*) \leftarrow \mathsf{Encap}(pub)$. Note that we can parse r^* as $r_1^* \| r_2^*$. Lastly, $\mathcal{B}$ sends the public parameters $(params, pub, cp)$ to $\mathcal{A}$.

Query Phase I. $\mathcal{B}$ initializes an empty table T and an empty set D. $\mathcal{A}$ can adaptively issue a polynomial number of the following queries. Note that on input a normal ciphertext of the form $\langle com^*, ct', tag, cm\rangle$, the decryption oracle (or the outsourcing decryption oracle) simply outputs $\perp$.

- *Private key query* $\langle \mathcal{S}\rangle$. $\mathcal{B}$ queries its own key generation oracle on $\mathcal{S}$ to obtain a private key $sk_{\mathcal{S}}$, which is given to $\mathcal{A}$. Additionally, $\mathcal{B}$ sets $D = D \cup \{\mathcal{S}\}$.
- *Transformation key query* $\langle \mathcal{S}\rangle$. If there exists an entry $(\mathcal{S}, tk_{\mathcal{S}})$ in table T, $\mathcal{B}$ obtains this entry. Otherwise, $\mathcal{B}$ queries its own transformation key generation oracle on $\mathcal{S}$ to obtain $tk_{\mathcal{S}}$, and stores in the table T the entry $(\mathcal{S}, tk_{\mathcal{S}})$. $tk_{\mathcal{S}}$ is given to $\mathcal{A}$ as the transformation key.
- *Decryption query* $\langle \mathcal{S}, ct\rangle$. When $\mathcal{A}$ submits $(\mathcal{S}, ct)$ where $ct = \langle com, ct', tag, cm\rangle$, $\mathcal{B}$ first checks whether $\mathsf{Verify}(ct, \mathcal{S}, S_{com}) = 1$ holds. If so, then $\mathcal{B}$ returns $\perp$. Otherwise $\mathcal{B}$ queries its own private key generation oracle on S_{com} and is given $sk_{S_{com}}$. Lastly, it returns the output of $\mathsf{Decrypt}(sk_{S_{com}}, ct)$ to $\mathcal{A}$.
- *Outsourcing decryption query* $\langle \mathcal{S}, ct, pct\rangle$. $\mathcal{B}$ searches in table T the entry $(\mathcal{S}, tk_{\mathcal{S}})$. If such an entry does not exist, it returns the failure symbol $\perp$. Otherwise, it checks whether $pct = \mathsf{Transform}_{out}(ct, tk_{\mathcal{S}})$ holds. If so, $\mathcal{B}$ returns the failure symbol $\perp$ as well. Otherwise, $\mathcal{B}$ queries the above decryption oracle on $(\mathcal{S}, ct)$ by itself and returns the output to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two l_1-length challenge messages m_0, m_1 and an access structure $\mathbb{A}^*$. $\mathcal{B}$ first flips a random coin $b \in \{0,1\}$ and sends $\mathcal{C}$ the messages $(M_0, M_1) = (m_b \| dec^*, 0^{l_1+l_3})$ and $\mathbb{A}'^*$. $\mathcal{C}$ flips a random coin $\beta \in \{0,1\}$, runs $ct'^* \leftarrow \mathsf{Encrypt}'(M_\beta, \mathbb{A}'^*)$ and sends ct'^* to $\mathcal{B}$. Lastly, $\mathcal{B}$ runs $tag^* \leftarrow \mathsf{Mac}_{r_1^*}(ct'^*)$ and $cm^* \leftarrow \mathsf{Commit}(m_b, r_2^*)$, and then sends $\mathcal{A}$ the challenge ciphertext $\langle com^*, ct'^*, tag^*, cm^*\rangle$.

Query Phase II. This phase is identical to Query Phase I, but with the restrictions defined in the security model.

Guess. The adversary $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0. Otherwise, it outputs 1.

It is a remarkable fact that, if the encryption query of $\mathcal{B}$ is answered with an encryption of $m_b \| dec^*$, then the view of $\mathcal{A}$ is exactly as in Game_1, otherwise the view of $\mathcal{A}$ is exactly as in Game_2. Thus, if $\mathcal{A}$ can distinguish Game_1 and Game_2 with a non-negligible advantage, we then can build an algorithm $\mathcal{B}$ that attacks the underlying CPA-secure ABE scheme with outsourced decryption with a non-negligible advantage. That is, $|\Pr_2[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]|$ is negligible.

Similarly, $|\Pr_2[\mathsf{Forge}] - \Pr_1[\mathsf{Forge}]|$ is negligible. The proof is extremely similar to the above proof. The only

difference is that $\mathcal{B}$ runs $\mathcal{A}$ to completion and then checks whether $\mathcal{A}$ has made any decryption query (or outsourcing decryption query), in which the normal ciphertext has the form $\langle com^*, ct', tag, cm\rangle$ and $\mathsf{Ver}_{r_1^*}(ct', tag) = 1$. If so, $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Game_3 is same as Game_2 except that the challenger chooses a random value $r = r_1 \| r_2$ uniformly from some proper domain, and sets the components tag^* and cm^* of the challenge ciphertext by running $tag^* \leftarrow \mathsf{Mac}_{r_1}(ct'^*)$ and $cm^* \leftarrow \mathsf{commit}(m_b, r_2)$, respectively. We show that $|\Pr_3[\mathsf{Succ}] - \Pr_2[\mathsf{Succ}]|$ is negligible. To see this, consider a simulator $\mathcal{B}$ breaking the hiding property of the underlying encapsulation scheme. $\mathcal{B}$ runs $\mathcal{A}$ to execute the following steps: first, the challenger runs $pub \leftarrow \mathsf{Init}(\lambda)$ and $(r_0, com^*, dec^*) \leftarrow \mathsf{Encap}(pub)$. It also selects a random value r_1 from some proper domain. It then chooses a random bit $\beta \in \{0,1\}$, and sends (pub, com^*, r_β) to $\mathcal{B}$. $\mathcal{B}$ then runs $(params, msk) \leftarrow \mathsf{Setup}'(1^\lambda, U \cup W)$, $cp \leftarrow \mathsf{InitCom}(1^\lambda)$ and sends the public parameters $(params, pub, cp)$ to $\mathcal{A}$. Since $\mathcal{B}$ has the master secret key msk, any query can be answered in the natural way. Note that on input a normal ciphertext of the form $\langle com^*, ct', tag, cm\rangle$, the decryption oracle (or the outsourcing decryption oracle) simply outputs $\perp$. When $\mathcal{A}$ submits two l_1-length challenge messages m_0, m_1 and an access structure $\mathbb{A}^*$, $\mathcal{B}$ flips a random coin $b \in \{0,1\}$ and runs $ct'^* \leftarrow \mathsf{Encrypt}'(0^{l_1+l_3}, \mathbb{A}'^*)$. It then parses r_β as $r_\beta = r_1 \| r_2$, and runs $tag^* \leftarrow \mathsf{Mac}_{r_1}(ct'^*)$ and $cm^* \leftarrow \mathsf{Commit}(m_b, r_2)$. The challenge ciphertext $ct^* = \langle com^*, ct'^*, tag^*, cm^*\rangle$. Further queries with the restrictions defined in the security model can be issued by $\mathcal{A}$ and then can be answered by $\mathcal{B}$ in the natural way. Lastly, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0, otherwise, it outputs 1. Obviously, if r_β is such that (r_β, com^*, dec^*) was output by $\mathsf{Init}(pub)$ then the view of $\mathcal{A}$ is exactly as in Game_2, while if r_β is chosen at random independently of com^* then the view of $\mathcal{A}$ is exactly as in Game_3. Hence, if $\mathcal{A}$ can distinguish Game_2 and Game_3 with a non-negligible advantage, we can build an algorithm $\mathcal{B}$ that attacks the hiding property of the underlying encapsulation scheme with a non-negligible advantage. That is, $|\Pr_3[\mathsf{Succ}] - \Pr_2[\mathsf{Succ}]|$ is negligible.

Note that event Forge in Game_3 is defined as before, but using the random key r_1. Similarly, $|\Pr_3[\mathsf{Forge}] - \Pr_2[\mathsf{Forge}]|$ is negligible. The proof is extremely similar to the above proof. The only difference is that $\mathcal{B}$ runs $\mathcal{A}$ to completion and then checks whether $\mathcal{A}$ has made any decryption or outsourcing decryption query, in which the normal ciphertext has the form $\langle com^*, ct', tag, cm\rangle$ and $\mathsf{Ver}_{r_1}(ct', tag) = 1$. If so, $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

We show that $\Pr_3[\mathsf{Forge}]$ is also negligible. In Game_3, we make a reasonable assumption that $\mathcal{A}$ is allowed to make at most q decryption queries and outsourcing decryption queries, where q is polynomial. We also assume that the normal ciphertext submitted in ith (outsourcing) decryption query is $\langle com_i, ct'_i, tag_i, cm_i\rangle$. We construct a simulator $\mathcal{B}$ breaking the underlying strong one-time message authentication code $\mathcal{MA}$. $\mathcal{B}$ first selects a random index $j \in [q]$ and then begins simulating Game_3 for $\mathcal{A}$ in the natural way. If the jth (outsourcing) decryption query occurs before the challenge phase, $\mathcal{B}$ halts and outputs $\langle ct'_j, tag_j\rangle$. Otherwise, when $\mathcal{A}$ submits two l_1-length challenge messages m_0, m_1 and an access structure $\mathbb{A}^*$, $\mathcal{B}$ flips a random coin $b \in \{0,1\}$ and selects $r = r_1 \| r_2$ uniformly at random from some proper domain. It then runs $ct'^* \leftarrow \mathsf{Encrypt}'(0^{l_1+l_3}, \mathbb{A}'^*)$. $\mathcal{B}$ submits ct'^* to its Mac oracle and is given tag^*. It also runs $cm^* \leftarrow \mathsf{Commit}(m_b, r_2)$. Next, $\mathcal{B}$ gives the challenge ciphertext $ct^* = \langle com^*, ct'^*, tag^*, cm^*\rangle$ to $\mathcal{A}$ and continues running $\mathcal{A}$ until the j^{th} (outsourcing) decryption query occurs. Then, $\mathcal{B}$ halts and outputs $\langle ct'_j, tag_j\rangle$. Note that the success probability of $\mathcal{B}$ in outputting a valid forgery is at least $\Pr_3[\mathsf{Forge}]/q$. Since $\mathcal{MA}$ is a strong one-time message authentication code and q is polynomial, then $\Pr_3[\mathsf{Forge}]$ is negligible. Hence, by using the triangle inequality principle, we obtain that $\Pr_1[\mathsf{Forge}]$ is negligible. Sequentially, we obtain that $|\Pr_1[\mathsf{Succ}] - \Pr_0[\mathsf{Succ}]|$ is negligible.

Game_4 is same as Game_3 except that the challenger sets the component cm^* of the challenge ciphertext by running $cm^* \leftarrow \mathsf{commit}(0^{l_1}, r_2)$. We then have $|\Pr_4[\mathsf{Succ}] - \Pr_3[\mathsf{Succ}]|$ is negligible. Due to the space limitation, we omit the proof, which is similar to the proof of Lemma 2. Moreover, in this game, since the challenge ciphertext is independent of the challenge messages chosen by the adversary, the adversary has no advantage. That is, $|\Pr_4[\mathsf{Succ}] - 1/2| = 0$.

Finally, by using the triangle inequality principle again, we have that $|\Pr_0[\mathsf{Succ}] - 1/2|$ is negligible. Therefore, we conclude that the adversary in the original IND-RCCA security game (Game_0) has a negligible advantage. This completes the proof of Theorem 3. □

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61133014, 61272413, 61272534, 61300226 and 61472165), the Guangdong Provincial Natural Science Foundation (S2013040014826), the Program for New Century Excellent Talents in University (NCET-12-0680), the Project of Science and Technology New Star of Guangzhou Pearl Rivel, the Fok Ying Tung Education Foundation (131066), and the Research Fund for the Doctoral Program of Higher Education of China (20134401110011, 20134401120017). J. Lai. and K. Chen are corresponding authors.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Int. Conf. Adv. Cryptol.*, 2005, pp. 457–473.

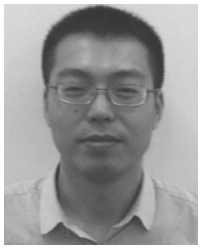
[2] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Proc. 8th Conf. Theory Cryptography*, 2011, pp. 253–273.

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

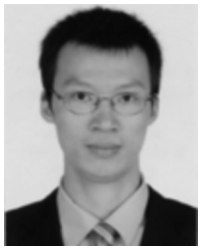
[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.

[5] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proc. 20th USENIX Conf. Security*, 2011, p. 34.

[6] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Proc. 19th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 1999, vol. 1666, pp. 537–554.
[7] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited (preliminary version)," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, 1998, pp. 209–218.
[8] J. Lai, R. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.
[9] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. 14th Int. Conf. Practice Theory Public Key Cryptography Conf. Public key Cryptography*, 2011, pp. 53–70.
[10] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro, "Generic constructions for chosen-ciphertext secure attribute based encryption," in *Proc. 14th Int. Conf. Public Key Cryptography*, 2011, pp. 71–89.
[11] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Adv. Cryptol.*, 1992, pp. 129–140.
[12] D. Boneh and J. Katz, "Improved efficiency for CCA-secure cryptosystems built using identity-based encryption," in *Proc. Int. Conf. Topics Cryptol.*, 2005, pp. 87–103.
[13] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptographic Eng.*, vol. 3, no. 2, pp. 111–128, 2013.
[14] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 195–203.
[15] L. Cheung, J. A. Cooley, R. Khazan, and C. Newport, "Collusion-resistant group key management using attribute-based encryption," *Group-Oriented Cryptographic Protocols*, p. 23, 2007.
[16] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. 29th Annu. Int. Conf. Theory Appl. Cryptographic Tech.*, 2010, pp. 62–91.
[17] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Proc. Int. Conf. Inf. Security Cryptol.*, 2009, pp. 20–36.
[18] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," in *Proc. 30th Annu. Conf. Adv. Cryptol.*, 2010, pp. 191–208.
[19] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Security*, 2010, pp. 261–270.
[20] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.
[21] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 735–737.
[22] S. Jahid, P. Mittal, and N. Borisov, "Easier: Encryption-based access control in social networks with efficient revocation," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Security*, 2011, pp. 411–415.
[23] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 515–534.
[24] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 121–130.
[25] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. 30th Annu. Int. Conf. Theory Appl. Cryptographic Tech.: Advances Cryptol.*, 2011, pp. 568–588.
[26] B. G. Kang, M. S. Lee, and J. H. Park. (2005). Efficient delegation of pairing computation, Cryptology ePrint Archive, Rep. 2005/259 [Online]. Available: http://eprint.iacr.org/
[27] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theoretical Comput. Sci.*, vol. 422, pp. 15–38, 2012.
[28] S. Hohenberger and B. Waters, "Attribute-based encryption with fast decryption," in *Proc. 16th Int. Conf. Practice Theory Public-Key Cryptography*, 2013, pp. 162–179.
[29] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.
[30] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on outsourced attribute-based encryption," in *Proc. 18th Eur. Symp. Res. Comput. Security*, 2013, pp. 592–609.
[31] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *Proc. 14th Int. Conf. Inf. Commun. Security*, 2012, pp. 191–201.
[32] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. 30th Annu. Conf. Adv. Cryptol.*, 2010, pp. 465–482.
[33] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. 30th Annu. Cryptol. Conf. Adv. Cryptol.*, 2010, pp. 483–501.
[34] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[35] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *Proc. 30th Annu. Int. Conf. Theory Appl. Cryptographic Tech.: Adv. Cryptol.*, 2011, pp. 129–148.
[36] R. Canetti, H. Krawczyk, and J. B. Nielsen, "Relaxing chosen-ciphertext security," in *Proc. 23rd Annu. Int. Cryptol. Conf.: Adv. Cryptol.*, 2003, pp. 565–582.



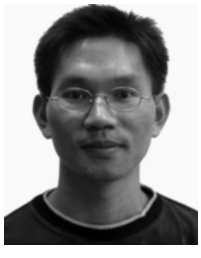
Xianping Mao received the MS degree in computer science and technology from China University of Mining and Technology, China, in 2007. Currently, he is working toward the PhD degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include cloud computing, information security and public-key cryptography.



Junzuo Lai received the PhD degree in computer science and technology from Shanghai Jiao Tong University, China, in 2010. From August 2008 to April 2013, he was a research fellow in Singapore Management University. Currently, he is a research professor in the School of Information Technology, Jinan University, China. His research interests include cryptography and information security.

Qixiang Mei received the PhD degree from Southwest Jiaotong University, China, in 2006. From March 2009 to April 2011, he was a postdoc in the State Key Laboratory of Information Security, China. Currently, he is an associate professor in the College of Information, Guangdong Ocean University, China. His research interests include information security and public-key cryptography.

Kefei Chen received the PhD degree from Justus Liebig University Giessen, Germany, in 1994. From 1996 to 2013, he was a professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Currently, he is a distinguished professor in the School of Science, Hangzhou Normal university, China. His research interests include cryptography and network security.



Jian Weng received the PhD degree in computer science and engineering from Shanghai Jiao Tong University, China, in 2008. From April 2008 to March 2010, he was a postdoc in the School of Information Systems, Singapore Management University. Currently, he is a professor and the vice dean in the School of Information Technology, Jinan University. His research interests include cryptography and information security.