This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

Verifiable and Exculpable Outsourced Attribute-Based Encryption for Access Control in Cloud Computing

Hui Ma*, Rui Zhang*, Zhiguo Wan, Yao Lu and Suqing Lin

Abstract—We propose two ciphertext-policy attribute-based key encapsulation mechanism (CP-AB-KEM) schemes that for the first time achieve both outsourced encryption and outsourced decryption in two system storage models and give corresponding security analysis. In our schemes, heavy computations are outsourced to Encryption Service Providers (ESPs) or Decryption Service Providers (DSPs), leaving only one modular exponentiation computation for the sender or the receiver. Moreover, we propose a general verification mechanism for a wide class of ciphertext-policy (cf. key-policy) AB-KEM schemes, which can check the correctness of the outsourced encryption and decryption efficiently. Concretely, we introduce a stronger version of verifiability (cf. [1]) and a new security notion for outsourced decryption called exculpability, which guarantees that a user cannot accuse DSP of returning incorrect results while it is not the case. With all these mechanisms, any dispute between a user and an outsource computation service provider can be easily resolved, furthermore, a service provider will be less motivated to give out wrong results. Finally, we implement our schemes in Charm [2], and the results indicate that the proposed schemes/mechanisms are efficient and practical.

1 Introduction

With the development of cloud computing, companies and individuals begin to outsource their data (often encrypted) to public clouds, thus flexible access control mechanism over this encrypted data became an important problem. As a cryptographic solution, Attribute-Based Encryption (ABE) was introduced by Sahai and Waters [3], which provides fine-grained (non-interactive) access control and encryption functionalities simultaneously.

However, a major drawback of ABE is that the heavy computational cost for encryption and decryption, largely the pairing and the exponentiation operations, which often grow with the complexity of access formula. This is a huge limitation on a mobile device (low battery and limited computational capability), since the resource consumption due to ABE computations will be tremendous, which essentially impedes ABE from wide-range deployment. To leverage computation burden on the user side, Outsourced ABE (OABE) [4], [5], [6], [7] was then proposed. In an OABE scheme, the complicated operations are outsourced to the cloud without revealing the private information and leaves only marginal computational cost to the user. E.g., in an electronic medical record (EMR) system, the tremendous medical records should be uploaded to public storages as soon as possible, then a doctor can download a new patient's whole medical records in real time. The medical records are encrypted by OABE, which protects the patients' privacy and achieves access control over encrypted records. A doctor can access medical records efficiently with a mobile device by outsourcing heavy operations to cloud servers.

OABE improves the efficiency of encryption and decryption, but the scenario becomes more complicated. On one hand, users need efficiency but do not wish public cloud servers to know private information. On the other hand, public cloud servers may be "lazy", and only compute partial operations for saving computation or bandwidth and return incorrect results. Incorrect decryption results could lead to a serious medical accident. In addition, it is necessary to have a mechanism that can exculpate cloud providers in a medical accident when they did return the correct results. In general, the following questions arise naturally:

1) How to prevent public cloud servers from learning private information?
2) How to guarantee the correctness of results returned by public cloud servers?
3) How to exculpate cloud providers from a false accusation?

1.1 Related Work

Attribute-Based Encryption was first introduced by Sahai and Waters under the name fuzzy identity-based encryption [3]. Goyal et al. [8] extended fuzzy IBE to ABE. Up to now, there are two forms of ABE: Key-Policy ABE (KP-ABE) [8], [9], [10], [11], where the key is assigned to an access policy and the ciphertext to a set of attributes, and Ciphertext-Policy ABE (CP-ABE) [12], [13], [14], where the ciphertext is assigned to an access policy and the key to a

- *Contact author.
- *This work was partially supported by the Foundation of Science and Technology on Information Assurance Laboratory (No. KJ-14-002), the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06010703), and the One Hundred Talents Project.*
Hui Ma, Rui Zhang and Suqing Lin are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. Hui Ma and Suqing Lin are also with the University of Chinese Academy of Sciences, Beijing 100049, China. Email: mahui@iie.ac.cn; r-zhang@iie.ac.cn; linsuqing@iie.ac.cn.
- *Zhiguo Wan is with School of Software, Tsinghua University. Email: wanzhiguo@gmail.com*
- *Yao Lu is with the University of Tokyo, Japan. Email: lywhhit@gmail.com*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

set of attributes. A user can decrypt a ciphertext only if the set of attributes satisfies the access policy.

Due to the complex paring and exponentiation operations in ABE, outsourcing complex operations to a cloud server becomes an efficient solution. In [15], Armknecht et al. proposed the notion of outsourced proofs of retrievability (OPOR), in which users can task an external auditor to perform and verify POR with the cloud provider. Pairing delegation has been presented in [16], [17], [18]. Green et al. [4] introduced outsourced decryption into ABE systems such that the complex operations during the decryption phase can be outsourced to a cloud server, only leaving one exponentiation operation for a user to recover the plaintext. Although they delegated complex operations to a cloud server, but they did not consider the correctness of results returned by the cloud.

In [7], Li et al. proposed securely OABE with checkability which supports both outsourced key-issuing and decryption. Lai et al. [1] formally introduced the verifiability of ABE and proposed an ABE scheme with verifiable outsourced decryption. Although [1] achieved the verifiability of outsourced decryption, the scheme doubled both the length of the ciphertext and the computational cost for encryption. In [19], Li et al. proposed a technique called "MapReduce", which outsourced complex operations for encryption to cloud servers. In [6], Li et al. proposed OABE that supported both outsourced key-issuing and decryption. We remark their work contributes to the widespread deployment of ABE. But the technique in [19], [6], [7] can be only applied to ABE schemes with access trees, but not for ABE schemes with Linear Secret Sharing Schemes (LSSS). In [20], Androulaki et al. proposed a time-based access control scheme, the access policy was not a formula but contained all the authorized users. In [21], Lai et al. proposed verifiable homomorphic encryption (VHE), which enables verifiable computation on outsourced encrypted data.

Recently, online/offline ABE [22] was proposed by Hohenberger and Waters (HW for short), which splits the original algorithm into two phases: an offline phase which does the majority of encryption computations before knowing the attributes/access control policy and generates an intermediate ciphertext, and an online phase which rapidly assembles an ABE ciphertext with the intermediate ciphertext after the attributes/access control policy is fixed. In [22], HW proposed two scenarios about the offline phase: 1) the user does the offline work on his smartphone. 2) A high-end trusted server helps the user with low-end device do the offline work.

We remark that the security of [22] is based on that the offline and online work is done by the entities who totally trust each other. It is not applicable to the scenario where the high-end server is not totally trusted by the user. Moreover, [1] doubles both the communication and computational cost, and does not consider exculpability. The verification for decryption in [7] shortens the length of the message significantly. To summarize, there is no VEOABE scheme that achieves both outsourced encryption and decryption.

1.2 Our Contribution

In this paper, we investigate all the above three questions, and come up with an affirmative answer: we introduce a new primitive, called verifiable and exculpable outsourced ABE (VEOABE). A VEOABE not only achieves the efficiency and privacy on the user side, but also brings strong verifiability to the outsourced computations. Our contribution is three-fold:

First, we present two CP-AB-KEM schemes that for the first time, can outsource computations of both encryption and decryptiont. Here, a major obstacle is that how to protect the randomness of ciphertext in encryption and prevent the leakage of the private key in decryption.

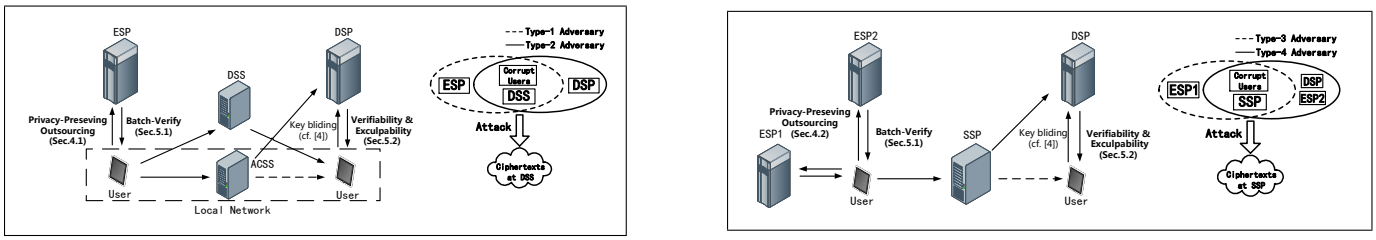
To remark, existing techniques cannot be easily adopted. E.g. as in [22], once the cloud which produces the intermediate ciphertext IT gets the corresponding ciphertext CT, it can easily recover the secret without the private key. Also, the technique in [19] is too not efficient and does not support LSSS [23], a popular and efficient tool in ABE setting. Here, we consider the following solutions: one is to randomize all the parameters and separate the ciphertexts from the intermediate ciphertexts, another is to introduce more randomness. Furthermore, to achieve secure outsourced decryption, we utilize an idea of key blinding [4]. As a result, the cost of both encryption or decryption on user side becomes merely one modular exponentiation!

Moreover, according to how the ciphertexts are stored, we classify the system model into "semi-open" or "open". In a semi-open system, there is a local network that is not accessed by adversaries and all the parameters are set by the user. In an open system, there is no local network but two different servers help to generate the intermediate ciphertexts (ITs), thus the system is secure if the two servers do not collude with each other. In general, extra entities will introduce more communication cost, such as the network latency and the battery consumption. However, the communication cost in our system is "imperceptible" for users, because the servers can generate ITs at any time without "knowing the messages", and mobile devices can download ITs when the devices are offline, say, while being charged.

Second, some cloud servers may become "lazy" if they cannot be detected, namely, they do nothing but return random results that "look similar" to the real ones, e.g., dummy strings of the same length. To deal with this issue, we propose general verification mechanisms to help users efficiently verify the computation results, therefore, the cloud servers should then be less motivated to be "lazy".

Concretely, during the encryption phase, we present a batch verification algorithm to check the correctness of outsourced computations for ABEs. During the decryption phase, we introduce two notions: verifiability and exculpability [15] for the purpose. Verifiability guarantees a user can efficiently check if the transformation is done correctly, as was considered in [1], [21]. However, our definition of verifiability is stronger. Exculpability is a novel requirement was not considered before: It guarantees that even a user who has the private key cannot accuse the decryption cloud server of outputting incorrect results while it was not the case. We give a concrete transform based on the key derivation function (KDF) [24] and the Pedersen commitment [25], which upgrades certain types of ABE scheme without verifiability or excludability to one that has. We note that these constructions are quite generic: it only requires underlying ABE satisfying some homomorphic property,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing



(a) The Semi-Open System Model

(b) The Open System Model

Fig. 1: Overview of System Models and Our Contributions

thus it is capable for both cipehrtext-policy ABE and key-policy ABE schemes.

Finally, we implement our CP-AB-KEM scheme and the verification mechanism within the *Charm* framework [2] on a laptop and evaluate its performance on a mobile phone. The performances of our scheme indicate that the high efficiency of our methodology.

Our contributions and an overview of the system model are summarized in Fig. 1. The arrows indicate data flows and the new techniques/notions introduced in the work are labeled in boldface.

2 Preliminary

In this section, we review some notations and definitions.

2.1 Notations

Let $A(a, b, ...) \rightarrow z$ denote the operation of running an algorithm A with inputs $a, b, ...$ and output z. If d is a string, let $|d|$ denote its length, while if S is a set, then $|S|$ denotes its size. Let $x||y$ denote the concatenation of two strings x and y. If a function f is negligible if for every $c > 0$ there exists $\lambda_0 > 0$ such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_0$.

Definition 1 (Bilinear Groups)**.** *Let* $\mathbf{G}$ *be an algorithm that takes as input a security parameter* λ *and outputs a tuple* $(p, \mathbb{G}, \mathbb{G}_T, e)$*, where* $\mathbb{G}$ *and* $\mathbb{G}_T$ *are multiplicative cyclic groups of prime order* p*, and* $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ *is a bilinear map such that: 1) Bilinearity: for all* $g, h \in \mathbb{G}$ *and* $a, b \in \mathbb{Z}_p^*$*, we have* $e(g^a, h^b) = e(g, h)^{ab}$*. 2) Nondegeneracy:* $e(g, h) \neq 1$ *whenever* $g, h \neq 1_{\mathbb{G}}$*.*

Definition 2 (Access Structure [23])**.** *Let* $\{P_1, ..., P_n\}$ *be a set of parties. A collection* $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}}$ *is monotone for* $\forall B$ *and* C*, if* $B \in \mathbb{A}, B \subseteq C$*, then* $C \in \mathbb{A}$*. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) of nonempty subsets of* $\{P_1, ..., P_n\}$*, i.e.,* $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}} \backslash \{\emptyset\}$*. The sets in* $\mathbb{A}$ *are called authorized sets, and the sets not in* $\mathbb{A}$ *are called unauthorized sets.*

Definition 3 (Linear Secret Sharing Schemes (LSSS) [23])**.** *A secret sharing scheme* Π *over a set of parties is called linear over* $\mathbb{Z}_p$ *if 1) The shares of the parties form a vector over* $\mathbb{Z}_p$*; 2) There exists a matrix* M *with* l *rows and* n *columns called the share-generating matrix for* Π*. There exists a function* ρ *which maps each row of the matrix to an associated party, i.e., for* $i = 1, ..., l$*, the value* $\rho(i)$ *is the party associated with row* i*. When we consider the column vector* $v = (s, r_2, ..., r_n)$*, where* $s \in \mathbb{Z}_p$ *is the secret to be shared, and* $r_2, ..., r_n \in \mathbb{Z}_p$ *are randomly chosen, then* Mv *is the vector of* l *shares of the secret* s *according to* Π*. The share* $(Mv)_i$ *belongs to party* $\rho(i)$*.*

3 The Model of VEOABE

In this section, we present the system models and the corresponding security definitions.

3.1 System Model

In an EMR system for a large hospital with several thousands of staffs, it is not economic to buy a high-performance device for each staff. In order to share the encrypted medical records via OABE, mobile devices with low battery and limited computational capability should outsource both the encryption and decryption computation to public cloud servers. However, the system model of OABE varies with the different circumstances of hospitals.

In this paper, we consider a semi-open system model and an open system model for OABE, both in the Key/Data Encapsulation Mechanism (KEM/DEM) setting [26]. There are two kinds of ciphertexts: the KEM ciphertexts that encapsulate session keys and the DEM ciphertexts encrypted by the session key. We introduce the following entities.

- ESP helps an user generate intermediate ciphertexts.
- DSP helps an user complete the decryption.
- SSP stores all the KEM/DEM ciphertexts.
- DSS stores all the DEM ciphertexts.
- ACSS stores all the KEM ciphertexts.

The overview of our system models and our contribution are illustrated in Fig. 1. The key issue during encryption is that ESP which produces the intermediate ciphertext (IT) should not get the corresponding ABE ciphertext. Once ESP accesses ABE ciphertexts, it can easily recover the secret.

In a semi-open system, we introduce a local network that is not accessed by adversaries to separate ABE ciphertexts from ESP. The user randomizes all the parameters chosen by ESP to prevent ESP from learning private information. In practice, the semi-open system is suitable for a hospital that has a reliable network that cannot be accessed by outsiders, and a local storage server (ACSS), which has storage capacity but poor computational capability.

In an open system, we remove the local network and adopt two different ESPs (ESP1, ESP2). So the final combined IT should be information-theoretically hidden from ESP1 or ESP2. The system is secure under the assumption that two ESPs do not collude with each other. Specifically, the ESPs can generate ITs at any time once they obtain PK,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

and when receiving a request from a user, the ESPs return ITs. Naturally, we can consider an extended open system, in which n different ESPs work together to provide encryption service (one server cannot collude with others at least. In this case, the collusion between user and at most $n-1$ malicious servers is allowed.

In practice, all the encryption and the decryption computation are outsourced to the public cloud servers, ESP1, ESP2, DSP and SSP. We remark that ESP2, DSP and SSP can be integrated into one service provider, e.g., Google, while ESP1 is another different service provider, e.g., Amazon. These companies have no reason to share their clients' information, even we can employ two different cloud providers that cannot reveal users' information by contract and do not know they work for a same client. Besides, it is hard to accurately locate the IT1 and IT2 in the same session from tremendous produced ITs.

During the decryption phase, we adopt key blinding [4]. In particular, a user first produces a conversion key and a retrieval key with a private key, then sends the conversion key and the request to DSP. DSP gives the user a transformed ciphertext, the user decrypts it with the retrieval key.

Let $\mathcal{S}$ represent a set of attributes, and $\mathbb{A}$ be an access structure. We define (I_{enc}, I_{key}) as the inputs to the encryption and key generation algorithm. In CP-AB-KEM, $(I_{enc}, I_{key}) = (\mathbb{A}, \mathcal{S})$, while in KP-AB-KEM, $(I_{enc}, I_{key}) = (\mathcal{S}, \mathbb{A})$. We define the function f as follows:

$$f(I_{key}, I_{enc}) = \begin{cases} 1 & if\ \ I_{enc} \in I_{key}\ in\ \text{KP-AB}\ setting \\ 1 & if\ \ I_{key} \in I_{enc}\ in\ \text{CP-AB}\ setting \\ 0 & otherwise. \end{cases}$$

Based on the proposed system models, the CP-AB-KEM with verifiable outsourced encryption and decryption for access structure space $\mathcal{G}$ consists of the following algorithms:

- $\mathbf{Setup}(\lambda, U) \rightarrow (PK, MSK)$. On input a security parameter λ and a universe description U, which defines a set of allowed attributes in the system, the algorithm outputs public parameters PK and a master secret key MSK.
- $\mathbf{KeyGen}(MSK, S) \rightarrow SK$. On input a master secret key MSK and a set of attributes S, the algorithm outputs a private key SK associated with S.
- $\mathbf{KeyGen.out}(SK) \rightarrow (TK, RK)$. On input a private key SK, the algorithm outputs a conversion key TK and the unique retrieval key RK.
- $\mathbf{Encrypt.out}(PK, N) \rightarrow IT$. On input public parameters PK and a number N which assumes a maximum bound of N rows in any LSSS access structure, the algorithm outputs an intermediate ciphertext IT.
- $\mathbf{Verify.enc}(PK, IT) \rightarrow b$. On input public parameters PK and an intermediate ciphertext IT, the algorithm outputs a bit $b \in \{0, 1\}$, $b = 1$ indicates ESP does the computations correctly, $b = 0$ indicates ESP does the computations incorrectly.
- $\mathbf{Encrypt.user}(PK, IT, (M, \rho)) \rightarrow (CT, key)$. On input public parameters PK, an intermediate ciphertext IT and an access structure (M, ρ), the algorithm outputs a ciphertext CT and an encapsulated key key.
- $\mathbf{Decrypt.out}(TK, CT) \rightarrow CT'$. On input a conversion key TK and a ciphertext CT, the algorithm outputs a transformed ciphertext CT'.
- $\mathbf{DecVerify.user}(RK, CT, CT') \rightarrow key$. On input a retrieval key RK, a ciphertext CT and a transformed ciphertext CT', the algorithm outputs an encapsulated key key.

For a fixed universe description U and $\lambda \in \mathbb{N}$, the CP-AB-KEM correctness property requires that for all $(PK, MSK) \in \mathbf{Setup}(\lambda, U)$, all $S \subseteq U$, all $\mathbb{A} \in \mathcal{G}$, all $SK \in \mathbf{KeyGen}(MSK, S)$, all $(TK, RK) \in \mathbf{KeyGen.out}(SK)$, all $IT \in \mathbf{Encrypt.out}(PK, N)$, all $(CT, key) \in \mathbf{Encrypt.user}(PK, IT, (M, \rho))$, all $CT' \in \mathbf{Decrypt.out}(TK, CT)$, if S satisfies $\mathbb{A}$, then $\mathbf{DecVerify.user}(RK, CT, CT')$ outputs key.

Practicality. Our technique adopts one or two servers for outsourced encryption. In general, extra entities will introduce more communication cost (battery consumption and network latency etc.), especially for mobile devices with low battery and limited computational capability. But in our system, the communication cost is offline, therefore, "imperceptible" to users, because the servers can generate ITs at any time once they get PK, and mobile devices can communicate with the servers when being charged.

For example, a mobile device is plugged into a power source when a doctor finishes the work for the day, then the device starts to communicate with the servers and do the preparation work. Next day, the doctor can rapidly perform ABE encryption operations in real time without significantly draining the battery and waiting for the servers' responses.

3.2 Security Definition

Adversarial Model. In a semi-open system, an ESP, a DSP and a DSS are three different public service providers, ACSS is a local trusted storage server that cannot be accessed by adversaries. We suppose that all the service providers except ACSS are "honest-but-curious" [19], [7]. More specifically, they follow the protocol but try to find out as much private information as possible. The channels between users and ESP/DSP are secure. The local network among the sender, ACSS and the receiver is not accessed by outsiders. We thus consider two types of adversaries:

- Type-1 adversary refers to some corrupted users colluding with ESP and DSS, who can obtain private keys of corrupted users, the ITs stored at ESP and the symmetric ciphertexts stored at DSS. It intends to decrypt uncorrupt users' ciphertexts at DSS.
- Type-2 adversary refers to some corrupted users colluding with DSS and DSP, who can obtain private keys of corrupted users, the symmetric ciphertexts stored at DSS, the ABE ciphertexts and the conversion keys stored at DSP. It intends to decrypt uncorrupt users' ciphertexts at DSS.

In an open system, two ESPs, a DSP and an SSP are four different public service providers. We suppose that all the providers are "honest-but-curious" [19], [7], and the two ESPs cannot collude with each other and the channels between users and ESP1/ESP2/DSP are secure. We thus consider two types of adversaries:

- Type-3 adversary refers to corrupted users colluding with one of ESPs (we denote ESP1) and SSP, who can obtain private keys of corrupted users, the ITs stored

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

at ESP1 and the ciphertexts stored at SSP. It intends to decrypt uncorrupt users' ciphertexts at SSP.

- Type-4 adversary refers to some corrupted users colluding with one of ESPs (we denote ESP2), DSP and SSP, who can obtain private keys of corrupted users, the ITs stored at ESP2, the conversion keys stored at DSP, the ciphertexts stored at SSP. It intends to decrypt uncorrupt users' ciphertexts at SSP.

Security. We consider RCCA (replayable chosen ciphertext attack) security [4], [27], which is "CCA" and allows "harmless" modifications to the ciphertext. According to different types of adversaries, RCCA security for CP-AB-KEM with outsourced encryption and decryption is described as follows. Since the security games are similar, and due to the limit of space, we postpone the games for type-1 and type-2 adversary to Appendix.

RCCA Security Game for Type-3 Adversary.

Setup. The challenger $\mathcal{C}$ runs *Setup* and gives PK to the adversary $\mathcal{A}$.

Phase 1. $\mathcal{C}$ initializes an empty table T, an empty set D and an integer counter $j = 0$. $\mathcal{A}$ can adaptively issue the following queries:

- $\mathbf{Create}(S)$: $\mathcal{C}$ sets $j = j + 1$. It runs *KeyGen* with S to obtain SK, then runs *Encrypt.out* with PK to obtain $IT1$. It stores the entry $(j, S, SK, IT1)$ in table T.
- $\mathbf{Corrupt.SK}(i)$: $\mathcal{C}$ checks whether the i^{th} entry (i, S, SK) exists in table T. If so, set $D = D \cup \{S\}$ and return SK, otherwise return $\perp$.
- $\mathbf{Corrupt.IT1}(i)$: $\mathcal{C}$ checks whether the i^{th} entry $(i, S, IT1)$ exists in table T. If so, return $IT1$, otherwise return $\perp$.
- $\mathbf{Decrypt}(i, CT)$: $\mathcal{C}$ checks whether the i^{th} entry $(i, S, SK, IT1)$ exists in table T. If so, return the output of the decryption on CT, otherwise return $\perp$.

Challenge. $\mathcal{A}$ gives a challenge value $\mathbb{A}^*$, with the restriction that for all $S \in D$, $f(S, \mathbb{A}^*) \neq 1$. $\mathcal{C}$ runs *Encrypt.out* two times to get $IT1^*$, $IT2^*$ and *Encrypt.user* to obtain (key^*, CT^*). $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$. If $b = 0$, it returns $(key^*, CT^*, IT1^*)$ to $\mathcal{A}$. If $b = 1$, it picks a random key R^* in the encapsulated key space and returns $(R^*, CT^*, IT1^*)$.

Phase 2. Phase 1 is repeated with the restrictions: 1) $\mathcal{A}$ cannot issue a private key query resulting in a value S which satisfies $f(S, \mathbb{A}^*) = 1$. 2) $\mathcal{A}$ cannot issue a decryption query resulting key^* or R^*.

Guess. $\mathcal{A}$ outputs a guess $b^{'}$ of b. $\mathcal{A}$ wins the game if and only if $b = b'$.

RCCA Security Game for Type-4 Adversary.

Setup, *Phase 2* and *Guess* are the same as the RCCA security game for type-3 adversary. *Phase 1* and *Challenge* are as follows.

Phase 1. $\mathcal{C}$ initializes an empty table T, an empty set D and an integer counter $j = 0$. $\mathcal{A}$ can adaptively issue the following queries:

- $\mathbf{Create}(S)$: $\mathcal{C}$ sets $j = j + 1$. It runs *KeyGen* with S to obtain SK, then runs *KeyGen.out* with SK to obtain TK, RK and runs Encrypt.out once with PK to obtain IT2. It stores the entry $(j, S, SK, TK, RK, IT2)$ in table T.
- $\mathbf{Corrupt.SK}(i)$: $\mathcal{C}$ checks whether the i^{th} entry (i, S, SK) exists in table T. If so, set $D = D \cup \{S\}$ and returns SK, otherwise return $\perp$.
- $\mathbf{Corrupt.TK}(i)$: $\mathcal{C}$ checks whether the i^{th} entry (i, S, TK) exists in table T. If so, return TK, otherwise return $\perp$.
- $\mathbf{Corrupt.IT2}(i)$: $\mathcal{C}$ checks whether the i^{th} entry $(i, S, IT2)$ exists in table T. If so, return $IT2$, otherwise, return $\perp$.
- $\mathbf{Decrypt}(i, CT)$: $\mathcal{C}$ checks whether the i^{th} entry (i, S, SK, TK, RK) exists in table T. If so, return the output of the decryption on CT, otherwise return $\perp$.

Challenge. It is the same as that in the game for type-3 adversary except that $IT1^*$ is replaced by $IT2^*$.

Definition 4. *A CP-AB-KEM scheme with outsourced encryption is RCCA-secure if all probabilistic polynomial time type-i adversaries ($i \in \{1, 3\}$) have at most a negligible advantage in the security games, denote:*

$$\epsilon_i = |\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda), \quad i \in \{1, 3\}.$$

Definition 5. *A CP-AB-KEM scheme with outsourced decryption is RCCA-secure if all probabilistic polynomial time type-i adversaries ($i \in \{2, 4\}$) have at most a negligible advantage in the security games, denote:*

$$\epsilon_i = |\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda), \quad i \in \{2, 4\}.$$

CPA Security. We say that a CP-AB-KEM scheme is *CPA-secure* (or secure against chosen-plaintext attacks) if we remove *Decrypt* oracle in both Phase 1 and 2.

Selective Security. We say that a CP-AB-KEM scheme is *selectively secure* if we add an *Init* stage before *Setup* where the adversary commits to the challenge access structure $\mathbb{A}^*$ at the beginning.

Verifiability of Outsourced Decryption. Verifiability guarantees that a user can efficiently check if the transformation is done correctly. It is described by a game $Game_V$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Our model defines stronger verifiability than [1]. Specifically, $\mathcal{C}$ gives $\mathcal{A}$ a valid private key rather than a valid ciphertext in [1]. If $\mathcal{A}$ can extract two different encapsulated keys from one ciphertext, it can break the verifiability. $Game_V$ consists of the following steps:

- **Setup.** $\mathcal{C}$ runs *Setup* and gives PK to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ gives a set of attributes $\mathcal{S}^*$, $\mathcal{C}$ runs *KeyGen (MSK, $\mathcal{S}^*$)* to obtain SK^* and sends it to $\mathcal{A}$.
- **Output.** $\mathcal{A}$ outputs an access structure $\mathbb{A}^*$ and a tuple $\{CT^*, TK^*, RK^*, CT_1^{*\prime}, CT_2^{*\prime}\}$, where $f(S^*, \mathbb{A}^*) = 1$, CT^* is an original ciphertext, TK^*, RK^* are the corresponding conversion key and retrieval key and $CT_1^{*\prime}, CT_2^{*\prime}$ are two transformed ciphertexts of CT^*. $\mathcal{A}$ wins the game if

$$\begin{aligned}
&(TK^*, RK^*) \leftarrow KeyGen.out(SK^*) \ \wedge \\
&DecVerify.user(RK^*, CT^*, CT_1^{*\prime}) \neq \perp \ \wedge \\
&DecVerify.user(RK^*, CT^*, CT_2^{*\prime}) \neq \perp \ \wedge \\
&DecVerify.user(RK^*, CT^*, CT_1^{*\prime}) \\
&\qquad \neq DecVerify.user(RK^*, CT^*, CT_2^{*\prime}).
\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

The advantage of the adversary in $Game_V$ is defined as $\Pr[\mathcal{A}\ wins\ Game_V]$.

Definition 6. *A CP-AB-KEM scheme with outsourced decryption is verifiable if all probabilistic polynomial time adversaries have at most a negligible advantage in the above game.*

$$\Pr[\mathcal{A}\ wins\ Game_V] \leq negl(\lambda).$$

Exculpability of Outsourced Decryption. Exculpability guarantees that a user who has the private key cannot accuse DSP of doing computations incorrectly when DSP actually returns correct results. It is described by a game $Game_E$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. $Game_E$ consists of the following steps:

- **Setup.** $\mathcal{C}$ runs *Setup* and gives PK to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ gives a set of attributes $\mathcal{S}^*$, $\mathcal{C}$ runs *KeyGen (MSK, $\mathcal{S}^*$)* to obtain SK^* and sends it to $\mathcal{A}$.
- **Output.** $\mathcal{A}$ outputs an access structure $\mathbb{A}^*$ and a tuple $\{CT^*, TK^*, CT^{*\prime}, RK_1^*, RK_2^*\}$, where $f(S^*, \mathbb{A}^*) = 1$, CT^* is an original ciphertext, TK^*, RK_1^*, RK_2^* are the conversion key and retrieval keys, and $CT^{*\prime}$ is the corresponding transformed ciphertext of CT^*. $\mathcal{A}$ wins the game if

$$\begin{aligned}
&CT^{*\prime} \leftarrow Decrypt.out(TK, CT^*)\ \wedge \\
&DecVerify.user(RK_1^*, CT^*, CT^{*\prime}) \neq \perp\ \wedge \\
&DecVerify.user(RK_2^*, CT^*, CT^{*\prime}) \neq \perp\ \wedge \\
&DecVerify.user(RK_1^*, CT^*, CT^{*\prime}) \\
&\qquad \neq DecVerify.user(RK_2^*, CT^*, CT^{*\prime}).
\end{aligned}$$

The advantage of the adversary in this game is defined as $\Pr[\mathcal{A}\ wins\ Game_E]$.

Definition 7. *A CP-AB-KEM scheme with outsourced decryption is exculpable if all probabilistic polynomial time adversaries have at most a negligible advantage in the above game.*

$$\Pr[\mathcal{A}\ wins\ Game_E] \leq negl(\lambda).$$

4 CP-ABE Schemes with Outsourced Encryption and Decryption

In this section, we present two OABE schemes with outsourced encryption and decryption and prove the security.

4.1 The Scheme for the Semi-open System

We present an OABE scheme with outsourced encryption and decryption in the semi-open system model and prove that it is selectively CPA-secure for type-1 and type-2 adversary in the standard model. Our scheme is based on an unbounded CP-ABE scheme by Rouselakis and Waters [28], which is proved to be selectively CPA-secure.

4.1.1 The Construction

Before presenting the scheme, we give some intuitions of our construction. If we straightforwardly apply HW online/offline ABE technique in [22], some private information and security parameters will be exposed when the offline work is done by an untrusted cloud server. To solve this problem, we first randomize all the parameters chosen by ESP and recompute the key in the user's encryption algorithm, then we introduce a local trusted server to store the ABE ciphertexts. This prevents ESP from learning the private information. Besides, we adopt the key blinding technique to achieve outsourced decryption.

Our first CP-AB-KEM scheme consists of 7 algorithms:

Setup(λ, U). The algorithm chooses a bilinear map $D = (\mathbb{G}, \mathbb{G}_T, e, p)$, where $p \in \Theta(2^\lambda)$ is the prime order of the groups $\mathbb{G}$ and $\mathbb{G}_T$. The attribute universe is consisting of elements in $\mathbb{Z}_p$. It chooses random generators $g, h, u, v, w \in \mathbb{G}$ and picks a random $\alpha \in \mathbb{Z}_p$. It sets the keys as

$$PK = (D, g, h, u, v, w, e(g,g)^\alpha) \quad MSK = (PK, \alpha).$$

KeyGen(MSK, S). On input a master secret key MSK and an attribute set $S = \{A_1, A_2, ..., A_k\} \subseteq \mathbb{Z}_p$, the algorithm picks $k+1$ random $r, r_1, r_2, ..., r_k \in \mathbb{Z}_p$ and computes $K_0 = g^\alpha w^r, K_1 = g^r$. Then for $i = 1$ to k, it computes

$$K_{i,2} = g^{r_i},\ K_{i,3} = (u^{A_i} h)^{r_i} v^{-r}.$$

It sets $SK = (S, PK, K_0, K_1, \{K_{i,2}, K_{i,3}\}_{i\in[1,k]})$.

KeyGen.out(SK). On input a private key SK, the algorithm chooses a random $\tau \in \mathbb{Z}_p^*$, then computes

$$K_0' = K_0^{1/\tau} = g^{\alpha/\tau} w^{r/\tau} \quad K_1' = K_1^{1/\tau} = g^{r/\tau}.$$

For $i = 1$ to k, it computes

$$K_{i,2}' = K_{i,2}^{1/\tau} = g^{r_i/\tau} \quad K_{i,3}' = K_{i,3}^{1/\tau} = (u^{A_i} h)^{r_i/\tau} v^{-r/\tau}.$$

The conversion key as TK is $(S, PK, K_0', K_1', \{K_{i,2}', K_{i,3}'\}_{i\in[1,k]})$, the retrieval key as RK is (PK, τ).

Encrypt.out(PK, N). On input public parameters PK and a maximum bound of N rows in any LSSS access structure, the algorithm first picks a random $s' \in \mathbb{Z}_p$ and computes $C_0 = g^{s'}$. Then for $j = 1$ to N, choose random $\lambda_j', x_j', t_j' \in \mathbb{Z}_p$ and compute

$$C_{j,1} = w^{\lambda_j'} v^{t_j'} \quad C_{j,2} = (u^{x_j'} h)^{-t_j'} \quad C_{j,3} = g^{t_j'}.$$

The intermediate ciphertext is $IT = (s', C_0, \{\lambda_j', t_j', x_j', C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[1,N]})$.

Encrypt.user(PK, IT, (M, ρ)). On input public parameters, an intermediate ciphertext IT, and an LSSS access structure (M, ρ), where M is an $l \times n$ matrix and $l \leq N$, the algorithm first picks a random $s \in \mathbb{Z}_p$ and computes $key = e(g,g)^{\alpha s}$, $C_7 = s - s'$. Then it picks random $y_2, ..., y_n \in Z_p$, sets the vector $\overrightarrow{y} = (s, y_2, ..., y_n)^T$ and computes a vector of shares of s as $(\lambda_1, ..., \lambda_l)^T = M\overrightarrow{y}$. For $j = 1$ to l, it picks a random $t_j \in \mathbb{Z}_p$, and computes

$$C_{j,4} = \lambda_j - \lambda_j' \quad C_{j,5} = x_j' t_j' - \rho(j) t_j \quad C_{j,6} = t_j - t_j'.$$

This will correct to the shares of s and the proper attribute $\rho(j)$, and randomize t_j. The encapsulated key is key. The dominant cost is two multiplications in $\mathbb{Z}_p$ per row of M and one modular exponentiation in $\mathbb{G}$. The ciphertext is $CT = ((M, \rho), C_0, C_7, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}, C_{j,6}\}_{j\in[1,l]})$.

Decrypt.out(TK,CT). On input a ciphertext $CT = ((M, \rho), C_0, C_7, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}, C_{j,6}\}_{j\in[1,l]})$ for access

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

structure (M, ρ) and a conversion key $TK = (S, PK, K'_0, K'_1, \{K'_{i,2}, K'_{i,3}\}_{i\in[1,k]})$ for attribute sets S, if S does not satisfy this access structure, the algorithm outputs $\perp$. Otherwise, it calculates $I = \{i : \rho(i) \in S\}$ and computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} \omega_i \cdot M_i = (1, 0, ..., 0)$, where M_i is the i-th row of the matrix M. Then it recovers the encapsulated key by computing

$$A = \prod_{i\in I}(e(C_{i,1} \cdot v^{C_{i,6}}, K'_1) \cdot e(C_{i,2} \cdot u^{C_{i,5}} \cdot h^{-C_{i,6}}, K'_{j,2}) \cdot e(C_{i,3} \cdot g^{C_{i,6}}, K'_{j,3}))^{\omega_i}$$

$$key' = \frac{e(C_0 \cdot g^{C_7}, K'_0)}{e(\omega^{\sum_{i\in I} C_{i,4}\omega_i}, K'_1) \cdot A} = e(g, g)^{\alpha s/\tau},$$

where j is the index of the attribute $\rho(i)$ in S (it depends on i). The transformed ciphertext is $CT' = ((M, \rho), e(g, g)^{\alpha s/\tau})$.

Decrypt.user(RK, CT′). On input a retrieval key RK and a transformed ciphertext CT', the algorithm computes

$$key = key'^{\tau} = e(g, g)^{(\alpha s/\tau)\cdot\tau} = e(g, g)^{\alpha s}.$$

4.1.2 Security Analysis

Theorem 1. *The CP-AB-KEM scheme in Sec. 4.1.1 is selectively CPA-secure with respect to Definition 4 ($i = 1$) and Definition 5 ($i = 2$) under the assumption that the scheme by RW [28] is a selectively CPA-secure CP-ABE scheme.*

The proof is similar to [8], and due to the limit of space, we postpone it to Appendix.

4.2 The Scheme for the Open System

We present an OABE scheme with outsourced encryption and decryption in the open system model and prove that it is selectively CPA-secure for type-3 and type-4 adversary in the standard model.

4.2.1 The Construction

We present another construction by utilizing two independent ESPs to hide all the parameters including the key during the encryption phase. Because the two ESPs cannot collude with each other and the distribution of the intermediate ciphertexts produced by each ESP is uniform in the IT space, the combined IT should be information-theoretically hidden from one of ESPs. This prevents ESPs from learning private information.

Our CP-AB-KEM scheme in an open system model is the same as the first CP-AB-KEM scheme but *Encrpyt.user* and *Decrypt.out*:

Encrypt.user(PK, IT1, IT2, (M, ρ)). On input the public parameters PK, two intermediate ciphertexts:

$$IT1 = (s', C'_0, \{\lambda'_j, t'_j, x'_j, C'_{j,1}, C'_{j,2}, C'_{j,3}\}_{j\in[1,N]})$$
$$IT2 = (s'', C''_0, \{\lambda''_j, t''_j, x''_j, C''_{j,1}, C''_{j,2}, C''_{j,3}\}_{j\in[1,N]}),$$

and an LSSS access structure(M, ρ), where M is an $l \times n$ matrix and $l \le N$, the algorithm first recomputes s and C_0,

$$s = s' + s'' \quad C_0 = C'_0 \cdot C''_0 = g^{s'} \cdot g^{s''} = g^{s'+s''}.$$

Then for $j = 1$ to N, recompute the following parameters:

$$\lambda_j = \lambda'_j + \lambda''_j \quad t_j = t'_j + t''_j \quad x_j = \frac{x'_j t'_j + x''_j t''_j}{t'_j + t''_j}$$

$$C_{j,1} = C'_{j,1} \cdot C''_{j,1} = \omega^{\lambda'_j} v^{t'_j} \cdot \omega^{\lambda''_j} v^{t''_j} = \omega^{\lambda'_j+\lambda''_j} v^{t'_j+t''_j}$$

$$C_{j,2} = C'_{j,2} \cdot C''_{j,2} = (u^{\frac{x'_j t'_j + x''_j t''_j}{t'_j + t''_j}} h)^{-(t'_j+t''_j)}$$

$$C_{j,3} = C'_{j,3} \cdot C''_{j,3} = g^{t'_j} \cdot g^{t''_j} = g^{t'_j+t''_j}.$$

The user obtains the combined intermediate ciphertext $IT = (s, C_0, \{\lambda_j, t_j, x_j, C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[1,N]})$.
Compute

$$key = e(g, g)^{\alpha s}.$$

Then pick random $y_2, ..., y_n \in Z_p$, set the vector $\overrightarrow{y} = (s, y_2, ..., y_n)^T$ and compute a vector of shares of s as $(\hat{\lambda_1}, ..., \hat{\lambda_l})^T = M\overrightarrow{y}$. For $j = 1$ to l, compute

$$C_{j,4} = \hat{\lambda_j} - \lambda_j \quad C_{j,5} = t_j(x_j - \rho(j)).$$

This will correct to the shares of s and the proper attribute $\rho(j)$. The user sets the ciphertext as :

$$CT = ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j\in[1,l]}).$$

The encapsulated key is key. The dominant cost is $4l + 1$ multiplications in $\mathbb{Z}_p$ and one modular exponentiation in $\mathbb{G}$.

Decrypt.out(TK,CT). On input a ciphertext $CT = ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j\in[1,l]})$ for access structure (M, ρ) and a private key $TK = (S, PK, K'_0, K'_1, \{K'_{i,2}, K'_{i,3}\}_{i\in[1,k]})$ for attribute sets S, if S does not satisfy this access structure, the algorithm outputs $\perp$. Otherwise, it calculates $I = \{i : \rho(i) \in S\}$ and computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} \omega_i \cdot M_i = (1, 0, ..., 0)$, where M_i is the i-th row of the matrix M. Then it recovers the encapsulated key by computing

$$A = \prod_{i\in I}(e(C_{i,1}, K'_1) \cdot e(C_{i,2} \cdot u^{C_{i,5}}, K'_{j,2}) \cdot e(C_{i,3}, K'_{j,3}))^{\omega_i}$$

$$key' = \frac{e(C_0, K'_0)}{e(\omega^{\sum_{i\in I} C_{i,4}\omega_i}, K'_1) \cdot A} = e(g, g)^{\alpha s/\tau},$$

where j is the index of the attribute $\rho(i)$ in S (it depends on i). The transformed ciphertext as $CT' = ((M, \rho), e(g, g)^{\alpha s/\tau})$.

4.2.2 Security Analysis

Theorem 2. *The CP-AB-KEM scheme in Sec. 4.2.1 is selectively CPA-secure with respect to Definition 4 ($i = 3$) and Definition 5 ($i = 4$) under the assumption that the scheme by RW [28] is a selectively CPA-secure CP-ABE scheme.*

Proof. For type-3 adversary $\mathcal{A}_3$ and type-4 adversary $\mathcal{A}_4$, *Corrupt.IT1* and *Corrupt.IT2* do not affect the security. Because the final IT^* is produced with $IT1^*$ and $IT2^*$. $\mathcal{A}_3$ and $\mathcal{A}_4$ can get one of $IT1^*$ and $IT2^*$, but cannot obtain any information of another, so the final IT^* should be information-theoretically hidden from $\mathcal{A}_3$ and $\mathcal{A}_4$.

To prove the security of our scheme with respect to Definition 5 ($i = 4$), we suppose that there exists a PPT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

adversary $\mathcal{A}_4$ who can win the CPA security game for type-4 adversary with a non-negligible advantage. We build a PPT simulator $\mathcal{B}$ to break the security of the RW scheme with a non-negligible advantage.

A simulator $\mathcal{B}$ interacts with $\mathcal{A}_4$ in the CPA security game for type-4 adversary with security parameter λ and the universe of attributes $U = \mathbb{Z}_p$.

- **Init.** $\mathcal{A}_4$ gives $\mathcal{B}$ its challenge access structure (M^*, ρ^*) (M^* is an $l \times n$ matrix), and $\mathcal{B}$ gives (M^*, ρ^*) to the RW challenger.
- **Setup.** $\mathcal{B}$ receives the public parameters $PK = (D, g, h, u, v, w, e(g,g)^{\alpha})$ from the RW challenger and gives them to $\mathcal{A}_4$.
- **Phase 1.** $\mathcal{B}$ initializes an empty table T and an integer counter $j = 0$. $\mathcal{A}_4$ can adaptively issue the following queries:
  - $\mathbf{Create}(S)$: $\mathcal{B}$ receives the private key query from $\mathcal{A}_4$, then sets $j = j + 1$ and passes it to the RW challenger to obtain SK, runs *KeyGen.out* with SK by itself to obtain TK, RK, runs *Encrypt.out* to obtain $IT2$. Then it stores the entry $(j, S, SK, TK, RK, IT2)$ in table T.
  - $\mathbf{Corrupt.SK}(i)$: $\mathcal{B}$ checks whether the i^{th} entry (i, S, SK) exists in table T. If so, it returns SK. Otherwise, it returns $\perp$.
  - $\mathbf{Corrupt.TK}(i)$: $\mathcal{B}$ checks whether the i^{th} entry (i, S, TK) exists in table T. If so, it returns TK. Otherwise, it returns $\perp$.
  - $\mathbf{Corrupt.IT2}(i)$: $\mathcal{B}$ checks whether the i^{th} entry $(i, S, IT2)$ exists in table T. If so, it returns $IT2$. Otherwise, it returns $\perp$.
- **Challenge.** $\mathcal{A}_4$ gives $\mathcal{B}$ its challenge access structure (M^*, ρ^*), $\mathcal{B}$ chooses two distinct random messages M_0, M_1 in the RW message space and sends them to RW challenger, and receives a challenge ciphertext $CT^*_{RW} = ((M^*, \rho^*), C, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[1,l]})$. Here C is the encrypted message times $e(g,g)^{\alpha s}$, $C_0 = g^s$ and for each row in M^*, $C_{j,1} = w^{\lambda_j} v^{t_j}, C_{j,2} = (u^{x_j} h)^{-t_j}, C_{j,3} = g^{t_j}$. Then it picks random $z_{1,1}, ..., z_{1,l}, z_{2,1}, ..., z_{2,l} \in \mathbb{Z}_p$, and computes the ciphertext $CT_{\text{type-4}}$ as $((M^*, \rho^*), C_0)$ followed by

$$C^*_{j,1} = C_{j,1} \cdot \omega^{-z_{1,j}} = \omega^{\lambda_j - z_{1,j}} v^{t_j},$$
$$C^*_{j,2} = C_{j,2} \cdot u^{-z_{2,j}} = (u^{\rho^*} h)^{t_j} \cdot u^{-z_{2,j}},$$
$$C^*_{j,3} = C_{j,3} = g^{t_j}, \quad C^*_{j,4} = z_{1,j}, \quad C^*_{j,5} = z_{2,j}.$$

  $\mathcal{B}$ runs *Encrypt.out* to obtain IT^*, selects a random bit $b_{\mathcal{B}} \in \{0,1\}$, computes $key_{guess} = C/M_{b_{\mathcal{B}}}$ and sends the tuple $(key_{guess}, CT_{\text{type-4}}, IT^*)$ to $\mathcal{A}_4$.
- **Phase 2.** B proceeds as in Phase 1.
- **Guess.** $\mathcal{A}_4$ outputs a bit $b_{\mathcal{A}_4}$. If $b_{\mathcal{A}_4} = 0$, it indicates that $\mathcal{A}_4$ guesses that key_{guess} is the key encapsulated by $CT_{\text{type-4}}$, then $\mathcal{B}$ outputs $b_{\mathcal{B}}$. If $b_{\mathcal{A}_4} = 1$, it indicates that $\mathcal{A}_4$ guesses that key_{guess} is a random key, then $\mathcal{B}$ outputs $1 - b_{\mathcal{B}}$. The distribution for $\mathcal{A}_4$ is perfect. So if $\mathcal{A}_4$ can win the CPA security game for type-4 adversary with a non-negligible advantage, then $\mathcal{B}$ can break the security of the RW scheme with the same advantage.

For $\mathcal{A}_3$, the proof is the same as above except that *Corrupt.TK()* is removed and *IT2* is replaced by *IT1*. □

5 The Verification Mechanism for VEOABE

In this section we first present a general verification mechanism. We first introduce batch verification for outsourced encryption, then give a verification mechanism achieving verifiability and exculpability in the outsourced decryption. Additionally, we give the security analysis.

5.1 Batch Verification for Outsourced Encryption

If a sender wants to encrypt message, it needs an IT produced by ESP. In case that ESP becomes "lazy" (it may not follow the agreed protocol, only execute a part of the computations to save its computation or bandwidth and return incorrect results), the sender cannot notice the incorrectness. Obviously, a trivial and inefficient solution is to let the sender check the results one by one. It's desirable that a sender can aggregate correctness checking operations, namely, batch verification. Based on the idea of [29], we design a function called *Verify.enc()* to check the correctness of IT. It includes a batch verification algorithm called *Batch-Verify*, which is used for checking the correctness of $C_{j,1}$, $C_{j,2}, C_{j,3}$. Apparently, *Verify.enc()* is much more efficient than the trivial solution.

Verify.enc(PK,IT). On input public parameters and an intermediate ciphertext $IT = (s', C_0, \{\lambda'_j, t'_j, x'_j, C_{j,1} = w^{\lambda'_j} v^{t'_j}, C_{j,2} = (u^{-t'_j x'_j} h^{-t'_j}), C_{j,3} = g^{t'_j}\}_{j\in[1,N]})$, the algorithm chooses a security parameter b. For $j = 1$ to N, compute $y'_j = x'_j \times (-t'_j) \bmod p$ and $F_j = C_{j,1} C_{j,2} C_{j,3}$. Verify whether $C_0 = g^{s'}$ or not, then runs *Batch-Verify*$(\{F_j, \lambda'_j, t'_j, y'_j, -t'_j, t'_j\}_{j\in[1,N]})$.

Algorithm 1 *Batch-Verify*

Input: $\mathbb{G}, p$ from a bilinear map $D = (\mathbb{G}, \mathbb{G}_T, e, p)$, five generators w, v, u, h, g of $\mathbb{G}$, $(F_1, \lambda_1, t_1, y_1, -t_1, t_1), ..., (F_n, \lambda_n, t_n, y_n, -t_n, t_n)$ with $\lambda_i, t_i, y_i, -t_i, t_i \in \mathbb{Z}_p$ and $F_i \in \mathbb{G}$, and a security parameter b.

Output: Check $\forall i \in \{1, ..., N\} : F_i = w^{\lambda_i} v^{t_i} u^{y_i} h^{-t_i} g^{t_i}$.

- Pick random $a_1, ..., a_N \in \{0,1\}^b$.
- Compute $A = \sum_{i=1}^{N} \lambda_i a_i \bmod p, B = \sum_{i=1}^{N} t_i a_i \bmod p, C = \sum_{i=1}^{N} y_i a_i \bmod p, D = -B, E = B$ and $R = \prod_{i=1}^{N} F_i^{a_i}$.
- If $R = w^A v^B u^C h^D g^E$ then accept, else reject.

We remark that the correctness of *Batch-Verify* is similar to [29], and it is applicable to most of the ABE schemes with modular exponentiations.

5.2 Verifiability and Exculpability of Outsourced Decryption

We present verifiability and exculpability for outsourced decryption and prove that the first scheme with outsourced decryption is selectively CPA-secure.

Obviously, this technique is a general construction and also applicable to the second scheme, so we just omit the construction and the proof due to the limit of space.

KeyGen, *KeyGen.out* and *Encrypt.out* are the same as the first scheme, we now introduce the remaining algorithms:

Setup(λ, U). It operates exactly as in the first CP-ABKEM besides that it first chooses another two random

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

generators $u', v' \in \mathbb{G}$, then adopts the key derivation function (KDF1) [24] with the output length $\mathcal{L}$, where $\mathcal{L} = |key| + |p|$. Finally, it chooses a deterministic collision-resistant hash function that maps strings uniformly to $\mathbb{Z}_p^*$, $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. It sets the keys as $PK = (D, g, h, u, v, w, u', v', e(g,g)^\alpha, KDF1, \mathcal{L}, H),\ MSK = (PK, \alpha)$.

Encrypt.user(PK, IT, (M, ρ)). It operates exactly as in the first CP-AB-KEM besides that it generates a new session key SSK with $key = e(g,g)^{\alpha s}$ and computes

$$KDF1(key, \mathcal{L}) = SSK \mid\mid d,\ \hat{C} = u'^{H(SSK)} v'^{H(d)}.$$

It sets the ciphertext as $CT = ((M,\rho), C_0, C_7, \hat{C}, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}, C_{j,6}\}_{j\in[1,l]})$. The encapsulated key is key and the session key is SSK. We add a Pedersen commitment $\hat{C}$ to achieve the verification of SSK.

Decrypt.out(TK,CT). It operates exactly as in the first CP-AB-KEM besides that its input is the ciphertext CT in *Encrypt.user* and it outputs $CT' = ((M,\rho), T_1 = \hat{C}, T_2 = e(g,g)^{\alpha s/\tau})$.

DecVerify.user(RK, CT, CT'). On input a retrieval key RK, a ciphertext CT and a transformed ciphertext CT', if $\hat{C} \neq T_1$, it outputs $\perp$, otherwise it computes the encapsulated key key and the session key SSK,

$$key = T_2^\tau = e(g,g)^{(\alpha s/\tau)\cdot\tau} = e(g,g)^{\alpha s}$$
$$KDF1(key, \mathcal{L}) = SSK \mid\mid d.$$

If $T_1 = u'^{H(SSK)} v'^{H(d)}$, it outputs key. Otherwise $\perp$.

Theorem 3. *The CP-AB-KEM scheme with verifiable and exculpable outsourced decryption is selectively CPA-secure with respect to Definition 5 ($i = 2$) under the assumption that the first CP-AB-KEM scheme is selectively CPA-secure, the KDF is secure, H is a secure deterministic collision-resistant hash function and the Pedersen commitment is computationally hiding.*

Proof. First, we consider the following 3 games:
$\mathbf{Game_0}$: The original CPA-secure game of CP-AB-KEM. $\hat{C}$ is generated as follows. $KDF1(key, \mathcal{L}) = SSK \mid\mid d$, $\hat{C} = u'^{H(SSK)} v'^{H(d)}$.
$\mathbf{Game_1}$: Same as $Game_0$ except that *KDF1()* is removed and $\hat{C} = u'^{R1} v'^{R2}$ where $R1$ and $R2$ are generated randomly.
$\mathbf{Game_2}$: Same as $Game_1$ except that $\hat{C} = u'^{0^{|R1|}} v'^{R2}$ where $R2$ is generated randomly.

Then we can prove the indistinguishability between the pairs $Game_0$ and $Game_1$, $Game_1$ and $Game_2$, respectively. Since the challenge ciphertext in $Game_2$ contains no information of the random value, we conclude that the advantage of the adversary in $Game_2$ is negligible. Thus the advantage of the adversary in the real game is negligible. We prove this theorem via the following lemma:

Lemma 1. *Suppose that the first CP-AB-KEM scheme is selectively CPA-secure, the KDF is secure and H is a secure (deterministic) collision-resistant hash function, then $Game_0$ and $Game_1$ are indistinguishable.*

Proof. Consider the following game:
$\mathbf{Game'_0}$: Same as $Game_0$ except that $KDF1(R, \mathcal{L}) = SSK' \mid\mid d'$, $\hat{C} = u'^{H(SSK')} v'^{H(d')}$ where R is generated randomly.

We first prove that $Game_0$ is indistinguishable from $Game'_0$. Suppose that there exits an adversary $\mathcal{A}$ who can distinguish $Game_0$ and $Game'_0$ with a non-negligible advantage, we can build a PPT simulator $\mathcal{B}$ to break the CPA-security of the first CP-AB-KEM scheme with a non-negligible advantage. A simulator $\mathcal{B}$ plays the role of the adversary in the first CP-AB-KEM scheme and interacts with $\mathcal{A}$ as follows:

- **Init.** $\mathcal{B}$ gives the challenge access structure (M^*, ρ^*) (M^* is an $l \times n$ matrix) from $\mathcal{A}$ to the challenger $\mathcal{C}$.
- **Setup.** When receiving the public parameters $PK = (D, g, h, u, v, w, e(g,g)^\alpha)$ from $\mathcal{C}$, $\mathcal{B}$ chooses random $x, y \in \mathbb{Z}_p^*$ and sets $u' = g^x$ and $v' = g^y$. $\mathcal{B}$ chooses the key derivation function (KDF1) with the output length $\mathcal{L}$ and a deterministic collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Then $\mathcal{B}$ gives $PK = (D, g, h, u, v, w, u', v', e(g,g)^\alpha, KDF1, \mathcal{L}, H)$ to $\mathcal{A}$.
- **Phase 1.** $\mathcal{B}$ forwards any queries from $\mathcal{A}$ to $\mathcal{C}$ and returns the replies to $\mathcal{A}$.
- **Challenge.** $\mathcal{B}$ forwards (M^*, ρ^*) from $\mathcal{A}$ to $\mathcal{C}$. $\mathcal{C}$ runs *Encrypt.out* to obtain IT^* and runs *Encrypt.user* to obtain (key^*, CT^*). $\mathcal{C}$ selects a random bit $b \in \{0,1\}$. If $b = 0$, it sets $K = key^*$. If $b = 1$, it picks a random key R^* in the encapsulated key space and sets $K = R^*$. Then it sends (K, CT^*) to $\mathcal{B}$. $\mathcal{B}$ computes $KDF1(K, \mathcal{L}) = SSK \mid\mid d, \hat{C} = u'^{H(SSK)} v'^{H(d)}$ and sends $(K, CT^*, \hat{C})$ to $\mathcal{A}$.
- **Phase 2.** $\mathcal{B}$ proceeds as in Phase 1.
- **Guess.** $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. $\mathcal{B}$ also outputs b' as its guess for b.

Apparently, if $b = 0$, $\mathcal{B}$ has properly simulated $Game_0$; otherwise, $\mathcal{B}$ has properly simulated $Game'_0$. If $\mathcal{A}$ can distinguish $Game_0$ and $Game'_0$ with a non-negligible advantage, we can build a simulator to break the CPA-security of the first CP-AB-KEM scheme with the same advantage. Then the proof of the indistinguishability between $Game_0$ and $Game'_0$ is completed.

Since the security of the KDF implies that $KDF(R, \mathcal{L})$ is indistinguishable from a random string, SSK' (or d') is indistinguishable from a random string. Since H is a deterministic collision-resistant hash function that maps strings uniformly to $\mathbb{Z}_p^*$, $H(SSK')$ (or $H(d')$) is indistinguishable from a random element in $\mathbb{Z}_p^*$. Thus $Game'_0$ and $Game_1$ are indistinguishable. Therefore, we conclude that $Game_0$ and $Game_1$ are indistinguishable. □

According to Lemma 1, SSK is indistinguishable from a random string, this guarantees the randomness of the new session key. $\hat{C}$ is the Pedersen commitment for a random value in $Game_1$. Since the Pedersen commitment is computationally hiding, $Game_1$ and $Game_2$ are indistinguishable. Thus $\hat{C}$ has no information of the random value. Combining all the discussions, we complete the proof of Theorem 3. □

Theorem 4. *Suppose that the DL assumption holds in the prime order bilinear group system, the above CP-AB-KEM scheme with outsourced decryption is verifiable and exculpable.*

Proof. To prove the verifiability and exculpability, we should show that it is secure against both two corresponding adversaries. The main difference between two types of adversaries is *Output*. Thus, we utilize a bit b to denote this

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

difference, i.e., if $b = 0$, the adversary can break the verifiability; if $b = 1$, the adversary can break the exculpability. We suppose that there exists a PPT adversary $\mathcal{A}$ who can win the game with a non-negligible advantage. We can build a PPT simulator $\mathcal{B}$ to solve the DL problem in the prime order bilinear group system with a non-negligible advantage.

$\mathcal{B}$ is given $(p, \mathbb{G}, \mathbb{G}_T, e, g, \beta = g^x)$ and wants to calculate $x = \log_g \beta$, so $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- **Setup.** $\mathcal{B}$ chooses random $\alpha, a, b, c, f, y, \in \mathbb{Z}_p^*$ and picks a key derivation function (KDF1) with the output length $\mathcal{L}$. $\mathcal{B}$ also chooses a deterministic collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Then $\mathcal{B}$ sets the public parameters as $PK = (p, \mathbb{G}, \mathbb{G}_T, e, g, h = g^a, u = g^b, v = g^c, w = g^f, u' = g^x, v' = g^y, e(g,g)^{\alpha s}, KDF1, \mathcal{L}, H)$. The master secret key is $MSK = (PK, \alpha)$. $\mathcal{B}$ sends PK to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ gives a set of attributes $\mathcal{S}^*$ to $\mathcal{B}$, $\mathcal{B}$ runs *KeyGen (MSK, $\mathcal{S}^*$)* to obtain SK^*, then sends it to $\mathcal{A}$.
- **Output.** $\mathcal{A}$ outputs an access structure $\mathbb{A}^*$ where $f(S^*, \mathbb{A}^*) = 1$. Then if $b = 0$, $\mathcal{A}$ outputs a tuple $\{CT^*, TK^*, RK^*, CT_1^{*\prime}, CT_2^{*\prime})\}$. If $b = 1$, $\mathcal{A}$ outputs $\{CT^*, TK^*, CT^{*\prime}, RK_1^*, RK_2^*)\}$.

If $b = 0$, $\mathcal{B}$ obtains $key^* = T_{1,2}^{\tau}$ and $key^{*\prime} = T_{2,2}^{\tau}$, where τ is RK^*. If $b = 1$, $\mathcal{B}$ obtains $key^* = T_2^{\tau_1}$ and $key^{*\prime} = T_2^{\tau_2}$, where τ_1 is RK_1^* and τ_2 is RK_2^*. Then computes $KDF1(key^*, \mathcal{L}) = SSK^*||d^*, KDF1(key^{*\prime}, \mathcal{L}) = SSK^{*\prime}||d^{*\prime}$, If $\mathcal{A}$ wins the game (means $key^* \neq key^{*\prime}$), then $\mathcal{B}$ can compute

$$g^{x \cdot H(SSK^*) + yH(d^*)} = u'^{H(SSK^*)} v'^{H(d^*)}$$
$$= T_1 = u'^{H(SSK^{*\prime})} v'^{H(d^{*\prime})} = g^{x \cdot H(SSK^{*\prime}) + yH(d^{*\prime})},$$

because of the property of KDF, with overwhelming probability, $SSK^* \neq SSK^{*\prime}$. Since H is a deterministic collision-resistant hash function, with overwhelming probability, $H(SSK^*) \neq H(SSK^{*\prime})$. $SSK^*, d^*, SSK^{*\prime}, d^{*\prime}, y, H$ are known by $\mathcal{B}$. $\mathcal{B}$ computes

$$x = \frac{y(H(d^{*\prime}) - H(d^*))}{H(SSK^*) - H(SSK^{*\prime})}$$

as the solution of the DL problem. □

Remark 1. *It is worth noting that the underlying AB-KEM is required to have homomorphic property which is compatible with the proposed verification mechanism, and our methodology in Sec. 5.1 and 5.2 is actually applicable to KP/CP-AB-KEM schemes.*

6 Performance Evaluations

In this section, we give both theoretical and experimental analyses of the CP-AB-KEM scheme (in Sec. 4.2.1) with the proposed verification mechanism (in Sec. 5).

6.1 Theoretical Analysis

Efficiency Comparison. Table 1 compares the efficiency of our schemes with those from [28], [22], [4], [20], [7], [1], [19]. "[22]+[4]" means the combination of HW online/offline ABE (in which a server does the offline work) and key blinding.

Our schemes are the first to achieve both verifiable outsourced encryption and decryption, and leave little computations to the user with limited resources. As shown in Table 1, during the encryption phase, the computational cost for the user is only 3 exponentiations (1 for encryption and 2 for verifiability) which is less than [28], [7], [1], [19], [20]. Compared with [19], our verification for outsourced encryption supports any LSSS access structure, which is a popular and efficient tool in ABE setting. HW online/offline ABE technique [22] has no exponentiations, but the outsource server in [22] is aware of all the privacy while the servers in our system know nothing valuable. During the decryption phase, the computational cost for the user is 3 exponentiations (1 for decryption and 2 for verifiability). Our verification mechanism for outsourced decryption requires nearly half of the exponentiations used in [1] for encryption and needs one less exponentiation in $\mathbb{G}_T$ than [1] for the final decryption phase. The computational cost of [7] is 1 exponentiation, because it directly appends the message to be encrypted with a dummy string 0^k. This shortens the length of message significantly. Compared with other schemes, our schemes have a considerable advantage in functionality and efficiency. We remark that they may find different applications in the future.

Communication Cost Comparison. Table 2 compares the communication costs of our schemes with that in [22], [4], [7], [1], [19]. For outsourced encryption, the communication cost of our schemes is less than [19], because in [19], a user should communicate with n servers and the communication cost is online, which is not suitable for mobile devices. Compared with [22], our techniques indeed increase the transmission length to achieve the privacy-preserving. But the communication cost is offline, which means it is "imperceptible" for users, because the servers can generate ITs at any time once they get PK, and all the communication cost arises in users' spare time. For outsourced decryption, compared with the original key blinding technique [4], [7], we just add an element of $\mathbb{G}$ to achieve the verifiability and exculpability of outsourced decryption. Compared with the first verification mechanism [1], our schemes needs one less element in $\mathbb{G}_T$. Besides, our schemes requires nearly half of the ciphertext length in [1], which greatly decreases the communication cost during the ciphertext transmission.

Taking a security parameter $k = 112$, the ciphertext size without verification ([22]+[4]) is 3259 $\sim$ 29629 bytes with N increasing from 10 to 100, while the additional bandwidth required for decryption verification is always 42 bytes.

6.2 Experimental Analysis

In order to evaluate the practical performance of our schemes, we implement the scheme in Sec. 4.2.1 and the verification mechanism for outsourced decryption in Sec. 5.2 (including the server side and the user side) within the *Charm* framework [2], which was developed to promote prototypes of cryptographic schemes. Our scheme uses 224-bit MNT elliptic curves from Pairing-Based Cryptography library [30] and is executed on an Intel Core i5-3320M CPU @2.6GHz and 4GB RAM running Ubuntu 14.04 LTS 64-bit system and Python 3.4.

Since *Charm* routines use asymmetric groups while the groups in our scheme are symmetric, only a small change needs to be made. Specifically, there are three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ and an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Because the time taken to execute operations in $\mathbb{G}_1$ is much less

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2499755, IEEE Transactions on Dependable and Secure Computing

TABLE 1: Efficiency Comparison

Schemes	Enc (Server)	Enc (USER)	Dec (Server)	Dec (USER)	Enc.Verify	Dec.Verify
[28]	×	(2+5l)Exp	×	|I|Exp+(3|I|+1)P	×	×
[22]	(2+5l)Exp	0Exp	×	(2|I|+1)Exp+(3|I|+2)P	×	×
[22]+[4]	(2+5l)Exp	0Exp	(2|I|+1)Exp+(3|I|+2)P	1Exp	×	×
[20]	×	6mExp	3Exp	2Exp	×	×
[7]	×	1P + (3+l)Exp	2|I|Exp+(2|I|+2)P	1Exp	×	√
[1]	×	(6l+6)Exp	2|I|Exp+(4|I|+2)P	4Exp	×	√
[19]	2nyExp	4Exp	×	|I|Exp+2|I|P	×	×
Our scheme 1	(1+5l)Exp	3Exp	(5|I|+2)Exp+(3|I|+2)P	3Exp	√	√
Our scheme 2	(2+10l)Exp	3Exp	(2|I|+1)Exp+(3|I|+2)P	3Exp	√	√

‡ Exp and P denote a modular exponentiation and a paring computation, respectively. I, n, l, y and m indicate the set that satisfies decryption requirement, the number of servers, the number of rows of the matrix M for LSSS, the number of leaf nodes and the number of users, respectively.

TABLE 2: Communication Cost Comparison

Schemes	Extra Servers (Enc)	Transfer Size (Enc)	Extra Servers (Dec)	Transfer Size (Dec)	Enc.Verify	Dec.Verify
[22]+[4]	1	(1+3l)$\|\mathbb{G}\|$+2$l\|\mathbb{Z}_p\|$	1	$\|\mathbb{A}\|$+1$\|\mathbb{G}_T\|$	×	×
[7]	×	×	1	$\|\mathbb{A}\|$+1$\|\mathbb{G}_T\|$	×	√
[1]	×	×	1	$\|\mathbb{A}\|$+1$\|\mathbb{G}\|$+2$\|\mathbb{G}_T\|$	×	√
[19]	n	n($\|\mathbb{A}\|$+2$y\|\mathbb{G}\|$)	×	×	×	×
Our scheme 1	1	(3+3l)$\|\mathbb{G}\|$+3$l\|\mathbb{Z}_p\|$	1	$\|\mathbb{A}\|$+1$\|\mathbb{G}\|$+1$\|\mathbb{G}_T\|$	√	√
Our scheme 2	2	2((2+3l)$\|\mathbb{G}\|$+2$l\|\mathbb{Z}_p\|$)	1	$\|\mathbb{A}\|$+1$\|\mathbb{G}\|$+1$\|\mathbb{G}_T\|$	√	√

‡ $|\mathbb{A}|$, $|\mathbb{Z}_p|$,$|\mathbb{G}|$ and $|\mathbb{G}_T|$ denote the size of an access structure, an element in $\mathbb{Z}_p$, an element in $\mathbb{G}$ and $\mathbb{G}_T$, respectively. n, l and y indicate the number of servers, the number of rows of the matrix M for LSSS, the number of leaf nodes, respectively.

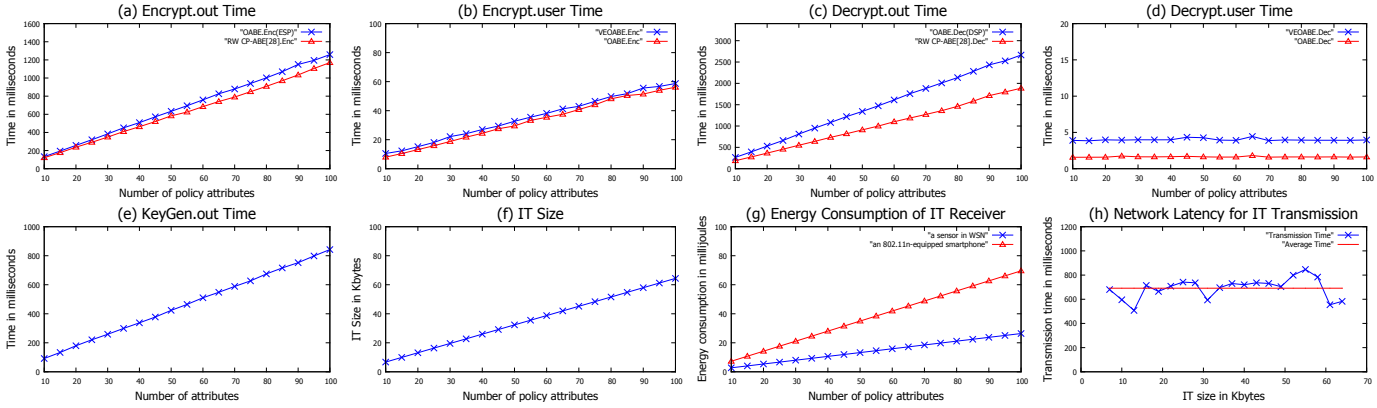


Fig. 2: Experimental Results

than $\mathbb{G}_2$ in MNT224 elliptic curve group, more operations in our scheme are executed in $\mathbb{G}_1$ rather than $\mathbb{G}_2$.

Experiment Setting. The complexity of access policy influences computational and communication cost in ABE. Accordingly, we generate access policies in the form of (S_1 and S_2 and ... and S_l) to simulate the worst situation, where S_i is an attribute. We set 20 distinct access policies in this form with N increasing from 10 to 100, and repeat each instance 10 times and take the average. We keep all the instances completely independent to each other. The time is given in milliseconds.

Computation Time. As depicted in Fig. 2, we show *Encrypt.out Time*, *Encrypt.user Time*, *Decrypt.out Time*, *Decrypt.user Time* and *KeyGen.out Time* of our scheme. To illustrate the efficiency of the verification mechanism, we also compare the verifiable scheme (VEOABE) with the unverifiable scheme (OABE). In Fig. 2(a), *Encrypt.out Time* is about 0.1s ∼ 1.2s, which is the time cost of an ESP, because the two ESPs can run in parallel in practice. In Fig. 2(b), despite *Encrypt.user* consists of combing the two ITs and generating verifiable ciphertexts, it still remains high efficiency (the time is about 10ms ∼ 58ms). We remark that the time increases with the complexity of the access policy, because the size of ITs combined by the user grows with the complexity of access policy. Compared with OABE in Fig. 2(b) and Fig. 2(d), it is obvious that our verification is remarkably efficient. Fig. 2(c) and Fig. 2(d) imply that DSP undertakes most of the encryption work, the user only takes about 4ms in VEOABE and about 1.6ms in OABE, which is highly efficient. In Fig. 2(e), *KeyGen.out Time* is about 91ms ∼ 842ms, which increases with the number of the attributes.

In Fig. 2(a) and Fig. 2(c), we present the performance of the original CP-ABE [28], in which a user should perform the entire encryption (Fig. 2(a)) and decryption (Fig. 2(c)) by himself. Compared with Fig. 2(b) and Fig. 2(d), our technique has an overwhelming advantage on the user side. However, in Fig. 2(a) and Fig. 2(c), the execution time of Encrypt.out (ESP) and Decrypt.out (DSP) is more than that

of [28]. For the encryption, the computation cost of ESP and the encryption algorithm in [28] is similar in Table 1, but the ESP needs to do extra preparation work. For the decryption, the computation cost of DSP is indeed more than that of the decryption algorithm in [28] in Table 1.

We also do experiments to evaluate the performance of outsourced decryption on a low-end smartphone. We execute C and Java on SAMSUNG GT-S6108 (an ARM-based 624MHz Marvell PXA918, 256MB RAM and Android 2.3.6). When doing the final decryption (for OABE/VEOABE), a user needs 1.6ms/3.9ms on the laptop while 78.9ms/155.1ms on GT-S6108. Interestingly, because our code utilizes no multi-threading, the execution time seems to be only related to the clock speed of CPU.

Power Consumption. To evaluate the battery consumption, we first refer to an often cited energy consumption model proposed by Heinzelman et al. [31], in which the receiver electronic $E_{\text{Rx-elec}} = 50\text{nj/bit}$. Then we refer to an energy consumption model proposed by Sun et al. [32], in which the average value of 802.11n per bit Rx energy consumption is 131.94 nJ/bit. Fig. 2(g) shows the battery consumption is 2.75mj $\sim$ 26.37mj for a sensor in WSN and 7.26mj $\sim$ 69.58mj for a smartphone, which all increases with the IT size.

Nevertheless, adopting two independent ESPs during the encryption phase may increase the communication cost, namely the battery consumption and the network latency. Our scheme actually shows a trade-off between the communication cost and the computation cost. A common sense in wireless sensor network (WSN) is that under the same conditions, the communication cost is more than the computation cost. So engineers may prefer to do more computations locally than transmit more information. But we still conclude that in most cases, our scheme is better because the ESPs can generate ITs at any time once they get PK and do not have to wait for the actual policy or messages are known. In particular, the computation of ITs (by ESPs) and a mobile device can download these ITs while it is offline, say, being charged. Then the actual online computations will be both fast and energy-efficient.

Network Latency. There are many factors causing network latency, such as, the transmission size, the network speed, Internet status etc., while the main factor of battery consumption is the transmission size. For outsourced encryption, Fig. 2(f) shows that the IT size for two ESPs is about 6.72KB $\sim$ 64.38KB, which increases with the complexity of the access policy. For outsourced decryption, the size of the transformed ciphertext is always 268B.

To evaluate the network latency, we do an experiment in a real network environment. We calculate the total network latency that a user successfully downloads ITs from a native public service provider (Baidu Cloud), including the request time, the response time and the transmission time. We repeat each instance (IT size is 7KB $\sim$ 64KB) 10 times and take the average value. Since a laptop also supports 802.11n, we use a laptop to evaluate the network latency. The laptop connects to Internet via Wi-Fi (802.11n) and the network speed is 8 Mbps. Fig. 2(h) shows that the total latency is nonlinear with IT size, e.g., the minimum time is 507ms and the maximal time is 845ms. This implies that the total latency is influenced by complex factors. We take the average latency 691ms as a benchmark.

In practice, the DSP can download ciphertexts directly from the SSP (as shown in Fig. 1) or the DSP and the SSP can be integrated into one service provider, the transmission between a user and the server is the transformed ciphertext with small fixed length 268B. The power consumption and the network latency of outsourced decryption can be estimated similarly as those of outsourced encryption.

Summary. Compared with non-outsourced scheme [28], our technique is practical and significantly enhances the user experience. Specifically, 1) the computation cost on the user side is tiny. 2) the extra communication cost of outsourced encryption is "imperceptible" for users. Besides, since the transmission size is small, the battery consumption and the network latency for online computations are better.

7 Discussion and Future Work

In [22], Hohenberger and Waters pointed out that online/offline ABE with outsourced decryption [4] is a good solution for mobile devices with high computational capability (e.g., capable of many modular exponentiations) when offline, e.g. being charged. However, in some cases, e.g., for a large hospital with several thousands of staffs, it is not economic to buy a high-performance device for each staff and our solution is more suitable.

Moreover, in the semi-open system model, it requires that a reliable network and a local server (ACSS) which cannot be accessed by outsiders. And in the open system model, two ESPs are introduced to achieve privacy-preserving, which makes the system model become complicated. In the future, it would be interesting to consider VEOABE with a simpler system model.

Acknowledgement

We thank anonymous reviewers for their valuable comments, and Yang Li for many helps on the experiments on mobile phones.

References

[1] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 8, pp. 1343–1354, 2013.

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

[3] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 457–473.

[4] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts." in *USENIX Security Symposium*, 2011, p. 3.

[5] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Proceedings of the 8th International Conference on Network and Service Management*. International Federation for Information Processing, 2012, pp. 37–45.

[6] 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 *Computer Security–ESORICS 2013*. Springer, 2013, pp. 592–609.

[7] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Transactions on Parallel and Distributed Systems*, p. 1, 2013.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 89–98.

[9] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 195–203.

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

[11] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," in *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 191–208.

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

[13] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[14] L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 456–465.

[15] F. Armknecht, J. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 831–843.

[16] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Smart Card Research and Advanced Application*. Springer, 2010, pp. 24–35.

[17] B. G. Kang, M. S. Lee, and J. H. Park, "Efficient delegation of pairing computation." *IACR Cryptology ePrint Archive*, vol. 2005, p. 259, 2005.

[18] P. P. Tsang, S. S. Chow, and S. W. Smith, "Batch pairing delegation," in *Advances in Information and Computer Security*. Springer, 2007, pp. 74–90.

[19] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *Information and Communications Security*. Springer, 2012, pp. 191–201.

[20] E. Androulaki, C. Soriente, L. Malisa, and S. Capkun, "Enforcing location and time-based access control on cloud-stored data," in *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, 2014, pp. 637–648.

[21] J. Lai, R. H. Deng, H. Pang, and J. Weng, "Verifiable computation on outsourced encrypted data," in *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I*, 2014, pp. 273–291.

[22] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography–PKC 2014*. Springer, 2014, pp. 293–310.

[23] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Technion-Israel Institute of technology, Faculty of computer science, 1996.

[24] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 631–648.

[25] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology–CRYPTO 1991*. Springer, 1992, pp. 129–140.

[26] V. Shoup, "A proposal for an iso standard for public key encryption (version 2.1)," *IACR E-Print Archive*, vol. 112, 2001.

[27] R. Canetti, H. Krawczyk, and J. B. Nielsen, "Relaxing chosen-ciphertext security," in *Advances in Cryptology–CRYPTO 2003*. Springer, 2003, pp. 565–582.

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

[29] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Advances in CryptologyłEUROCRYPT'98*. Springer, 1998, pp. 236–250.

[30] B. Lynn, "The stanford pairing based crypto library," http://crypto.stanford.edu/pbc.

[31] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*. IEEE, 2000, pp. 10–pp.

[32] L. Sun, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "Modeling wifi active power/energy consumption in smartphones," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 41–51.

Hui Ma received his B.E. degree in information security from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2008. He is currently pursuing the Ph.D. degree in information security with the the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He is currently working on the security mechanisms in cloud computing.

Rui Zhang received his B.E. degree from Tsinghua University, and M.S./PhD. degrees from the University of Tokyo, respectively. He was a JSPS research fellow before he joined AIST, Japan as a research scientist. Now he is with Institute of Information Engineering (IIE), Chinese Academy of Sciences as a research professor. His research interests include applied cryptography, network security and information theory.

Zhiguo Wan is currently a Lecturer with the School of Software, Tsinghua University, Beijing, China. His main research interests include security protocols, privacy enhancing techniques, and system security. He received the B.S. degree in computer science from Tsinghua University, in 2002, and the Ph.D. degree from the School of Computing, National University of Singapore, Singapore, in 2007.

Yao Lu received his B.S. degree from Harbin Institute of Technology in 2009, and Ph.D. degree from the the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2015. His research interests include cryptography and information security, in particular, public-key cryptosystem design and analysis.

Suqing Lin received her B.S. degree from Wenzhou Normal College, China, in 2003 and M.S. degree from Sichuan normal University, China, in 2006. She was a teacher with City College of Wenzhou University from 2006 to 2015. Her primary research interest lies in the field of cryptography and information security .