Dual Access Control for Cloud-Based Data Storage and Sharing

Jianting Ning, Xinyi Huang, Willy Susilo, *Senior Member, IEEE,* Kaitai Liang, *Member, IEEE*, Ximeng Liu *Member, IEEE*, and Yinghui Zhang, *Member, IEEE*

Abstract—Cloud-based data storage service has drawn increasing interests from both academic and industry in the recent years due to its efficient and low cost management. Since it provides services in an open network, it is urgent for service providers to make use of secure data storage and sharing mechanism to ensure data confidentiality and service user privacy. To protect sensitive data from being compromised, the most widely used method is encryption. However, simply encrypting data (e.g., via AES) cannot fully address the practical need of data management. Besides, an effective access control over download request also needs to be considered so that Economic Denial of Sustainability (EDoS) attacks cannot be launched to hinder users from enjoying service. In this paper, we consider the *dual access control*, in the context of cloud-based storage, in the sense that we design a control mechanism over both data access and download request without loss of security and efficiency. Two dual access control systems are designed in this paper, where each of them is for a distinct designed setting. The security and experimental analysis for the systems are also presented.

Index Terms—Cloud-based data sharing, access control, cloud storage service, Intel SGX, attribute-based encryption.

1 Introduction

In the recent decades, cloud-based storage service has attracted considerable attention from both academia and industries. It may be widely used in many Internet-based commercial applications (e.g., Apple iCould) due to its long-list benefits including access flexibility and free of local data management. Increasing number of individuals and companies nowadays prefer to outsource their data to remote cloud in such a way that they may reduce the cost of upgrading their local data management facilities/devices. However, the worry of security breach over outsourced data may be one of the main obstacles hindering Internet users from widely using cloud-based storage service.

In many practical applications, outsourced data may need to be further shared with others. For example, a Dropbox user Alice may share photos with her friends. Without using data encryption, prior to sharing the photos, Alice needs to generate a sharing link and further share the link with friends. Although guaranteeing some level of access control over unauthorized users (e.g., those are not Alice's friends), the sharing link may be visible within the Dropbox administration level (e.g., administrator could reach the link).

Since the cloud (which is deployed in an open network) is not be fully trusted, it is generally recommended to encrypt the data prior to being uploaded to the cloud to ensure data security and privacy. One of the corresponding solutions is to directly employ an encryption technique (e.g., AES) on the outsourced data before uploading to cloud, so that only specified cloud user (with valid decryption key) can gain access to the data via valid decryption.

To prevent shared photos being accessed by the "insiders" of the system, a straightforward way is to designate the group of authorized data users prior to encrypting the data. In some cases, nonetheless, Alice may have no idea about who the photo receivers/users are going to be. It is possible that Alice only has knowledge of attributes w.r.t. photo receivers. In this case, traditional public key encryption (e.g., Paillier Encryption), which requires the encryptor to know who the data receiver is in advance, cannot be leveraged. Providing policy-based encryption mechanism over the outsourced photos is therefore desirable, so that Alice makes use of the mechanism to define access policy over the encrypted photos to guarantee only a group of authorized users is able to access the photos.

In a cloud-based storage service, there exists a common attack that is well-known as *resource-exhaustion attack*. Since a (public) cloud may not have any control over download request (namely, a service user may send unlimited numbers of download request to cloud server), a malicious service user may launch the denial-of-service (DoS)/distributed denial-of-service (DDoS) attacks to consume the resource of cloud storage service server so that the cloud service could not be able to respond honest users' service requests. As a result, in the "pay-as-you-go" model, economic aspects could be disrupted due to higher resource usage. The costs of cloud service users will rise dramatically as the attacks

J. Ning is with the School of Mathematics and Computer Science, Fujian Normal University, China, and the School of Information Systems, Singapore Management University, Singapore. (e-mail: jtning88@gmail.com).
X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, China. (e-mail: xyhuang81@gmail.com)
W. Susilo is with the Institute of Cybersecurity and Cryptology, University of Wollongong, Australia (e-mail: wsusilo@uow.edu.au).
K. Liang is with the Department of Computer Science, University of Surrey, U.K. (e-mail: k.liang@surrey.ac.uk).
X. Liu is with College of Mathematics and Computer Science, Fuzhou University, China. (e-mail: snbnix@gmail.com).
Y. Zhang is with the National Engineering Laboratory for Wireless Security, Xi'an University of Posts & Telecommunications, China, and the School of Information Systems, Singapore Management University, Singapore. (e-mail: yhzhaang@163.com).

scale up. This has been known as Economic Denial of Sustainability (EDoS) attack [32], [33], which targets to the cloud adopter's economic resources. Apart from economic loss, unlimited download itself could open a window for network attackers to observe the encrypted download data that may lead to some potential information leakage (e.g., file size). Therefore, an effective control over download request for outsourced (encrypted) data is also needed.

In this paper, we propose a new mechanism, dubbed *dual access control*, to tackle the above aforementioned two problems. To secure data in cloud-based storage service, attribute-based encryption (ABE) [9] is one of the promising candidates that enables the confidentiality of outsourced data as well as fine-grained control over the outsourced data. In particular, Ciphertext-Policy ABE (CP-ABE) [5] provides an effective way of data encryption such that access policies, defining the access privilege of potential data receivers, can be specified over encrypted data. Note that we consider the use of CP-ABE in our mechanism in this paper. Nevertheless, simply employing CP-ABE technique is not sufficient to design an elegant mechanism guaranteeing the control of both data access and download request.

A strawman solution to the control of download request is to leverage dummy ciphertexts to verify data receiver's decryption rights. It, concretely, requires data owner, say Alice, to upload multiple "testing" ciphertexts along with the "real" encryption of data to cloud, where the "testing" ciphertexts are the encryptions of dummy messages under the same access policy as that of the "real" data. After receiving a download request from a user, say Bob, cloud asks Bob to randomly decrypt one of the "testing" ciphertexts. If a correct result/decryption is returned (i.e. indicating Bob is with valid decryption rights), Bob is authorized by Alice to access the "real" data, so that the cloud allows Bob to download the corresponding ciphertext.

Nevertheless, several disadvantages of the above approach may be identified as follows. First of all, the data owner, Alice, is required to encrypt a number of dummy ciphertexts under the same policy as the "real" ciphertext. This may yield a considerable computational overhead for Alice, which may bring inconvenience in practice, for example, Alice just wants to upload one photo to iCloud from her cellphone, but needs to prepare more than one ciphertexts. Second, all ciphertexts, including dummy ones, are uploaded to cloud at the same time. This inevitably imposes extra cost on network bandwidth (as well as prolonging data uploading time), which may not be applicable to some service users whose cellular network is under pay-as-you-go plan or equipped with old generation of broadband cellular network technology (e.g., 3G). Third, a data receiver/user, Bob, has to additionally decrypt a random-chosen "testing" ciphertext from cloud, as a test of his valid download request. As a result, Bob has to "pay" double (decryption price) for accessing to the "real" data, which again may not be scalable in resource constrained setting. Therefore, this paper raises the following question:

"Does there exist a cloud-based mechanism supporting dual access control (over both fine-grained data access and download request) without loss of security and efficiency?"

1.1 Our Results and Contributions

We answer the aforementioned question affirmatively by presenting two secure and efficient cloud-based dual access control systems[1] in different contexts. With the aim of providing an efficient way of dual access control, we briefly introduce the technical roadmap as follows. To guarantee the confidentiality of outsourced data without loss of policy-based access control, we start with a CP-ABE system [36], which is seen as one of the building blocks. We further employ an effective control over data users' download request on the top of the CP-ABE system. We design a new approach to avoid using the technique of "testing" ciphertext. Specifically, we allow data user to generate a download request. Upon receiving the download request, with help of the authority or the enclave of Intel SGX, a cloud server is able to check if the data user is authorized to gain access to the data. No other information is revealed to the cloud server except the knowledge of whether the user is authorized. Based on the above mechanism, the cloud maintains the control of the download request. The systems we propose are with the following distinct features:

(1) Confidentiality of outsourced data. In our proposed systems, the outsourced data is encrypted prior to being uploaded to cloud. No one can access them without valid access rights.

(2) Anonymity of data sharing. Given an outsourced data, cloud server cannot identify data owner, so that the anonymity of owner can be guaranteed in data storage and sharing.

(3) Fine-grained access control over outsourced (encrypted) data. Data owner keeps controlling his encrypted data via access policy after uploading the data to cloud. In particular, a data owner can encrypt his outsourced data under a specified access policy such that only a group of authorized data users, matching the access policy, can access the data.

(4) Control over anonymous download request and EDoS attacks resistance. A cloud server is able to control the download request issued by any system user, where the download request can set to be anonymous. With the control over download request, we state that our systems are resistant to EDoS attacks.

(5) High efficiency. Our proposed systems are built on the top of the CP-ABE system [36]. Compared with [36], they do not incur significant additional computation and communication overhead. This makes the systems feasible for real-world applications.

1.2 Related Work

To apply fine-grained policy-based control over encrypted data, ABE [9], [29] has been introduced in the literature. Concretely, ABE has two main research branches: one is CP-ABE, and the other is KP-ABE which refers to as key-policy ABE. This paper mainly deals with the former. In a CP-ABE, decryption key is associated with attribute set and ciphertext is embedded with access policy. This feature makes CP-ABE quite suitable for secure cloud data sharing

1. Hereafter, we use "dual access control" to denote the control over encrypted data and download request.

(compared to KP-ABE). Note this is so because KP-ABE requires decryption key to be associated with access policy which yields heavy storage cost for cloud user. Since the introduction of seminal CP-ABE [9], many works have been proposed to employ CP-ABE in various applications, e.g., accountable and traceable CP-ABE [22], [23], [24], [25], multi-authority [10], [17], outsourced CP-ABE [15], [16], [21], and extendable variants [34].

Although being able to support fine-grained data access, CP-ABE, acting as a single solution, is far from practical and effective to hold against EDoS attack [11] which is the case of DDoS in the cloud setting [11], [39]. Several countermeasures to the attack [12], [33] have been proposed in the literature. But Xue et al. [38] stated that the previous works could not fully defend the EDoS attack in the algorithmic (or protocol) level, and they further proposed a solution to secure cloud data sharing from the attack. However, [38] suffers from two disadvantages. First, the data owner is required to generate a set of challenge ciphertexts in order to resist the attack, which enhances its computational burden. Second, a data user is required to decrypt one of the challenge ciphertexts as a test, which costs a plenty of expensive operations (e.g., pairing). Here the computational complexity of both parties is inevitably increased and meanwhile, high network bandwidth is required for the delivery of ciphertexts. The considerable computational power of cloud is not fully considered in [38]. In this paper, we will present a new solution that requires less computation and communication cost to stand still in front of the EDoS attack. Recently, Antonis Michalas [20] proposed a data sharing protocol that combines symmetric searchable encryption and ABE, which allows users to directly search over encrypted data. To implement the functionality of key revocation in ABE, the protocol utilizes SGX to host a revocation authority. Bakas and Michalas [3] later extended the protocol in [20] and proposed a hybrid encryption scheme that reduces the problem of multi-user data sharing to that of a single-user. In particular, the symmetric key used for data encryption is stored in an SGX enclave, which is encrypted with an ABE scheme. Similar to [20], it deals with the revocation problem in the context of ABE by employing the SGX enclave. In this work, we employ SGX to enable the control of the download request (such that the DDoS/EDoS attacks can be prevented). In this sense, the purpose and the technique of ours are different from that of the protocols in [3], [20].

1.3 Organization

The rest of the paper is organized as follows. Section 2 introduces the necessary preliminaries. The system architecture and security model are given in Section 3. The proposed systems and security analysis are in Section 4 and 5, respectively. Performance evaluations are provided in Section 6. Finally, Section 7 presents a brief conclusion and future work.

2 Background

2.1 Notation

Let PPT be probabilistic polynomial-time. Define $[k] = \{1, 2, ..., k\}$ for $k \in \mathbb{N}$. Let $(a_1, a_2, ..., a_n)$ be a row vector and $(a_1, a_2, ..., a_n)^{\perp}$ be a column vector. By v_i we denote the i-th element in a vector $\vec{v}$. Let $\mathcal{G} = (G, G_T, p, e)$ be the groups and the bilinear mapping description, where G and G_T are two multiplicative cyclic groups of prime order p and $e : G \times G \to G_T$ is a bilinear map.

2.2 Prime Order Bilinear Groups.

Let G and G_T be two multiplicative cyclic groups of prime order p. Let g be a generator of G and $e : G \times G \to G_T$ be a bilinear map. The bilinear map e has the following properties: (1) Bilinearity: $\forall u, v \in G$ and $x, y \in \mathbb{Z}_p$, we have $e(u^x, v^y) = e(u, v)^{xy}$; (2) Non-degeneracy: $e(g, g) \neq 1$. We say that G is a bilinear group if the group operations in G and the bilinear map $e : G \times G \to G_T$ can both be computed efficiently.

2.3 Complexity Assumption

Assumption 1. *(Decisional q-parallel Bilinear Diffie-Hellman Exponent assumption (decisional q-Parallel BDHE) [36]) The Decisional q-parallel Bilinear Diffie-Hellman Exponent problem as follows. Initially choose a group G of prime order p according to the security parameter, pick a random group element $g \in G$, and $q + 2$ random exponents $c, s, b_1, b_2, ..., b_q \in \mathbb{Z}_p$. If an adversary is given the group description (p, G, G_T, e) and $\vec{z}$ including the following terms:*

$$
\begin{array}{ll}
g, g^s & \\
g^c, ..., g^{c^q}, g^{c^{q+2}}, ..., g^{c^{2q}} & \\
g^{s \cdot b_j}, g^{c/b_j}, ..., g^{(c^q/b_j)}, g^{(c^{q+2}/b_j)}, ..., g^{(c^{2q}/b_j)} & \forall_{1 \leq j \leq q} \\
g^{c \cdot s \cdot b_k / b_j}, ..., g^{(c^q \cdot s \cdot b_k / b_j)} & \forall_{1 \leq j,k \leq q, k \neq j}
\end{array}
$$

it is hard for the adversary to distinguish $e(g, g)^{sc^{q+1}} \in G_T$ from an element R which is randomly chosen from G_T. An algorithm $\mathcal{A}$ that outputs $\beta \in \{0, 1\}$ has advantage ϵ in solving the above assumption if $|\Pr[\mathcal{A}(\vec{z}, e(g, g)^{sc^{q+1}}) = 0] - \Pr[\mathcal{A}(\vec{z}, R) = 0]| \geq \epsilon$.

Definition 1. *We say that the decisional q-Parallel BDHE assumption holds if no PPT algorithm has a non-negligible advantage in solving the decisional q-Parallel BDHE problem.*

2.4 Ciphertext-Policy Attribute-based-Encryption

Ciphertext-Policy Attribute-based-Encryption (CP-ABE) is a versatile encryption supporting fine-grained access control over encrypted data. In a CP-ABE system, each data user is issued with a secret key according to his attributes. A data owner can choose an access structure $\mathbb{A}$ and encrypt his data under $\mathbb{A}$. The encrypted file can be decrypted by any data user whose attribute set satisfies $\mathbb{A}$. CP-ABE systems proposed in recent years usually make essential use of linear secret-sharing schemes. The definitions of access structure and linear secret-sharing schemes are shown as follows.

Access Structure [4], [25]: Let S denote an attribute universe. A collection $\mathbb{A} \subseteq 2^S$ is called monotone if $\forall B, C \in \mathbb{A}$: $if\ B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. A collection (respectively, monotone collection) $\mathbb{A} \subseteq 2^S$ of non-empty subsets of S is an access structure (respectively, monotone access structure) on S. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Linear Secret-Sharing Schemes (LSSS) [4], [25]): Let S be an attribute universe and p be a prime. A secret-sharing scheme $\prod$ over S is called linear (over $\mathbb{Z}_p$) if (1) The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$; (2) For each access structure $\mathbb{A}$ on S, there exists a matrix M with l rows and n columns called the share-generating matrix for $\prod$. For $i = 1, ..., l$, we define a function ρ labels row i of M with attribute $\rho(i)$ from S. When we consider the column vector $\vec{v} = (s, r_2, ..., r_n)$, 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 $M\vec{v} \in \mathbb{Z}_p^{l \times 1}$ is the vector of l shares of the secret s according to $\prod$. The share $(M\vec{v})_j$ "belongs" to attribute $\rho(j)$ for $j \in [l]$.

A CP-ABE system consists of four algorithms the following four algorithms:

- **Setup($\lambda, \mathcal{U}$).** The setup algorithm takes as input a security parameter λ and attribute universe $\mathcal{U}$, and outputs a master secret key MSK and the public parameters PP.
- **Encrypt($PP, \mathbb{A}, M$).** The encryption algorithm takes as input the public parameters PP, an access structure $\mathbb{A}$ and a message M, and outputs a ciphertext CT.
- **KeyGen(MSK, S).** The key generation algorithm takes as input the master secret key MSK and an attribute set S, and outputs a secret key SK.
- **Decrypt(PP, SK, CT).** The decryption algorithm takes as input the public parameters PP, a secret key SK and a ciphertext CT. If the attribute set of SK satisfies the access structure of CT, it outputs a message M; otherwise, it outputs $\perp$.

The definition of CP-ABE's security can be found in [36], which achieves indistinguishability under chosen-plaintext attacks (i.e., is IND-CPA secure).

2.5 Authenticated Encryption with Associated Data

Authenticated encryption with associated data (AEAD) is a form of symmetric-key encryption which simultaneously provides confidentiality as well as integrity [28]. A symmetric-key encryption scheme **SE** mainly consists of the following two PPT algorithms:

- **SE.Enc**$(m, sk) \rightarrow ct$: On input a message m and a symmetric key sk, it outputs a ciphertext ct.
- **SE.Dec**$(ct, sk) \rightarrow m$: On input a symmetric key sk and a ciphertext ct, it outputs a message m.

An symmetric-key encryption scheme **SE** should be semantically secure under a chosen plaintext attack.

2.6 Intel SGX

Intel Software Guard Extensions (SGX) is a set of new instructions available on recent-model Intel CPUs that allow for the creation of isolated execution environments called enclaves [19]. Our systems build on the notion of enclave, which is designed to run code and handle secrets in a trustworthy manner, even on a host where the system memory and OS are untrusted. The enclave provides three main security properties: isolation, sealing, and attestation. *Isolation* restricts access to a hardware guarded area of memory such that only that particular enclave can access it. Any other process on the same processor, even the OS, hypervisor, cannot access that memory. *Sealing* provides a way of encrypting enclave secrets for persistent storage to disk such that the secrets can be retrieved even if the enclave is torn down. Encryption is performed using a private seal key that is unique to that particular enclave, no process other than the exact same enclave can decrypt (or modify) it. *Attestation* enables an entity to verify that the desired code is indeed running securely and unmodified within the enclave. In particular, there are two forms of attestation: *local attestation* and *remote attestation* [7]. Local attestation is used for attestation between two enclaves on the same platform. The two enclaves on the same machine can derive a shared key, called *Report Key*, using the Root Seal Key shared between them. Remote attestation enables an enclave to generate a report that can be verified by any remote entity. Specifically, in order to generate a *quote*, an enclave first attests to a special enclave called the Quoting Enclave locally and sends it a report. After verifying the received report, the Quoting Enclave converts it into a quote, which contains the same underlying data. Essentially, the quote is signed with a secret key for an anonymous group signature scheme called Intel Enhanced Privacy ID (EPID) [7], [13]. The signature generated from EPID can be essentially verified by using the group public key.

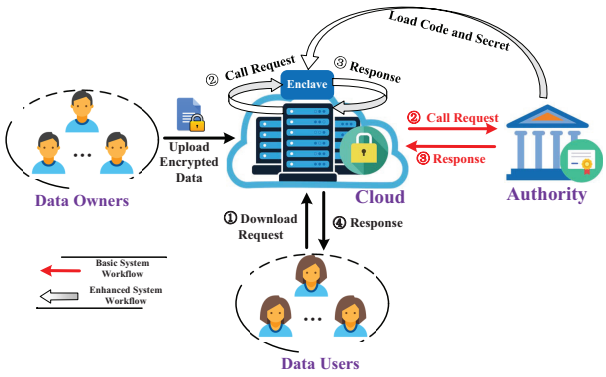


Fig. 1 Overview of system architecture

3 System Architecture and Security Model

3.1 System Architecture

The architectures of our dual access control systems for cloud data sharing are shown in Fig. 1. Concretely, the systems consist of the following entities:

- **Authority** is responsible for initializing system parameters and data user registration. Also, it handles the call request from the cloud in the first proposed construction.
- **Data owner** holds the data and wants to outsource his data to the cloud. In particular, data owners (only) want to share their data with those who satisfy certain conditions (e.g., professors or associate professors). They will be *offline* once their data have been uploaded to the cloud.
- **Data user** wants to download and decrypt the encrypted data shared in the cloud. Those who are authorized can download the encrypted file and further decrypt it to access the plaintext.

- **Cloud** provides convenient storage service for data owners and data users. Specifically, it stores the outsourced data from data users and handles the download requests sent by data users.
- **Enclave** handles the call request from the cloud (used in the second system).

The description of workflow is introduced as follows. Data owners encrypt their data under the access policies chosen by themselves and upload the encrypted data to the cloud. Authorized data users can download the shared data by sending a download request to the cloud. Upon receiving a download request from an authorized data user (see ① in Fig. 1), the cloud does as follows.

(a) For our basic system, the cloud sends a call request to the authority (see red ② between the cloud and the authority in Fig. 1). After receiving a response from the authority (see red ③ between the cloud and the authority in Fig. 1), the cloud sends a response back to the data user (see ④ in Fig. 1).

(b) For our enhanced system, the cloud sends a call request to the enclave (see black ② above the cloud in Fig. 1). After receiving a response from the enclave (see black ③ above the cloud in Fig. 1), the cloud sends a response back to the data user (see ④ in Fig. 1).

3.2 Security Assumptions

The security assumption of each entity is described as follows.

- **Authority** is *fully trusted* by other entities.
- **Data owner** is *honest* in the sense that she/he encrypts the outsourced data and uploads the encrypted data to the cloud honestly.
- **Data user** is *malicious* in the sense that she/he may try to download the shared file which is not authorized for her/him and launch the EDoS attacks.
- **Cloud** is *honest-but-curious* in the sense that it may gather sensitive information curiously by observing the transcript but will not deviate from the specification. Specifically, it will store the outsourced data and handles the access control on the download request honestly. However, it may try to infer more information (they are not supposed to know) than what is revealed by the transcript.
- **Enclave** is *fully trusted* in the sense that it will execute the loaded program (using the loaded secret data inside if necessary) honestly[2]. In particular, the program and static data inside the enclave cannot be read or modified from the outside, even for `root` nor any other type of special-access program. It is a hardware-based guarantee provided by the Software Guard extensions (SGX).

3.3 Design Goals and Security Requirements

Given the security assumption of each entity described above, the main design goals of our proposed systems include the following:

- **Anonymous data sharing.** The identity of the data owner should not be public. In particular, for a newly uploaded file, the real identity of the file's owner cannot be identified by the cloud.
- **Confidentiality of shared data.** The data outsourced to the cloud should be invisible to the cloud and unauthorized data users.
- **Anonymous download request.** For a download request sent from a data user, the request should be anonymous in the sense that the cloud cannot identify who sends this request.
- **Access control on download request.** To thwart a malicious data user's EDoS attacks, the shared data in the cloud can only be download by those who are authorized.
- **Access control on shared data.** The shared data can only be decrypted by those who are authorized.

Based on the security assumptions and design goals presented above, the security requirements of our systems include:

- **Security against *honest-but-curious* cloud:** a) The cloud cannot identify the owner of any newly uploaded file; b) The cloud cannot obtain the plaintext of the encrypted data stored on it; c) The cloud cannot identify the sender of any download request.
- **Security against *malicious* data user:** a) Any unauthorized data user cannot download the shared file(s) (i.e., resistant to data user's EDoS attacks); b) Any unauthorized data user cannot decrypt the shared file if the data user obtains the file. A data user is defined to be unauthorized if his/her attribute set does not satisfy the access policy of shared file.

4 The Proposed Systems

4.1 System overview

We employ the use of a hybrid system to protect the data, which combines the efficiency of a symmetric-key system with the convenience of a public-key system. In particular, the proposed dual access control systems are both in Key/Data Encapsulation Mechanism (KEM/DEM) setting [31]. The message is encrypted by an efficient symmetric-key encryption scheme, while the inefficient public-key scheme (i.e., the CP-ABE) is used only to encrypt/decrypt a short key value.

To achieve the security requirements of anonymous data sharing, confidentiality of shared data and access control on shared data, we employ the CP-ABE technique as the basic building block. Specifically, we present the construction based on the CP-ABE scheme in [36] due to its efficiency and elegant construction. To achieve the security requirements of anonymous download request and access control on download request, we design an effective mechanism that the cloud can judge whether a data user is authorized or not without revealing any sensitive information (including the identity of the data user, the plaintext of the outsourced data) to it. In the first system, the cloud needs the help of the authority during the judgement on the download request (sent by a data user). As a result, the authority needs to be always *online*. However, in some other cases in practice,

2. In our implementation, we will utilize the SGX SDK cryptographic library and add the property of *data-oblivious* to make it to be secure against side channel attack. In this sense, the enclave can be assumed to be fully trusted.

the authority may not be always online. This leads to the second (enhanced) system where the authority can be offline after the parameter initialization procedure. In particular, we employ the SGX technique to replace the role of the authority during the access control on download request procedure.

We now explain the rationale behind our proposed systems. In order to provide strong security and privacy guarantees for shared data on the cloud (that could defend the EDoS attack), a cloud-based data sharing system should support dual access control as described in Section 1. We start from the CP-ABE system proposed in [36], and adapt it to the KEM/DEM setting. However, simply employing the CP-ABE construction from [36] in the KEM/DEM setting is not sufficient to provide dual access control. New technique needs to be introduced such that the control of both data access and download request can be guaranteed. Different from the strawman solution described in Section 1, we introduce a new approach to avoid using the "testing" ciphertext in the strawman solution. Specifically, we allow the data owner to generate a download request, which contains a randomized form of the secret key held by the data owner. The download request retains the "decryption capability" of the secret key such that it can be used to test whether the underlying data owner is capable to to decrypt the shared ciphertext(s). Since the above mentioned component contained in the download request is randomized, it cannot be utilized to infer the owner of the secret key. That is, the download request enables the cloud to check whether the data owner of the download request is authorized without leaking the identity of the underlying data owner (i.e., the download request is anonymous). To further prevent leaking secret information to the cloud, the verification of download request needs the help of the authority or the enclave of Intel SGX. Our first system is designed for the case where the verification of download request involves the help of the authority, while the second system is designed for the case where the enclave of Intel SGX is involved during the verification of download request procedure. We note that our technique described above is general in the sense that it can be applied to most of the current CP-ABE constructions based on bilinear maps.

4.2 The Basic System

The procedures are described as follows. Let Γ be the CP-ABE scheme in [36].

- Parameter Initialization :
  In this procedure, the authority setups the system and publishes public parameters which will be utilized by other entities. It is run by the authority, consisting of the following steps:
  - Initialize a security parameter λ and obtain a bilinear map group system (G, G_T, p, e) by calling the group generator $\mathcal{G}$ with λ as input, where G, G_T are groups with prime order p and e is a bilinear map.
  - Randomly choose $\mathcal{U}$ group elements $\{A_1, ..., A_{\mathcal{U}}\} \in G$ as the attribute universe of the system and $\alpha, a \in \mathbb{Z}_p$.
  - Publish the parameters $PK = (g, g^a, A_1, ..., A_{\mathcal{U}}, e(g,g)^\alpha)$ and keep the parameters $MSK = (a, g^\alpha)$ as the master secret key.

  Note that the generation of PK, MSK is the almost same with the underlying CP-ABE scheme in [36], excepting that MSK contains an additional parameter a. That is, the above procedure operates the same as the algorithm **Setup**$(\lambda, \mathcal{U})$ of Γ, excepting that it adds a into the master secret key MSK.

- Data User Registration :
  The authority takes charge of issuing secret access credential for each registered data user in this procedure. In order to join the system, each data user needs to register with the authority. For each data user, the registration procedure (run by the authority) consists of the following steps:
  - Assign a unique attribute set S to the data user.
  - Randomly choose $v \in \mathbb{Z}_p$, set the secret key
    $$SK = (L_1 = g^\alpha g^{av}, L_2 = g^v, \{L_{3,x} = A_x^v\}_{\forall x \in S}, S).$$
    Note that this step operates the same as the algorithm **KeyGen**(MSK, S) of Γ.
  - Issue SK to the data user.

- Shared File Generation and Outsourcing :
  In this procedure, data owners encrypt their data under the access policies chosen by themselves and upload the encrypted data to the cloud. It is run by the data owner, consisting of the following steps:
  - Encrypt the message M as follows: choose a random symmetric key $\mathcal{SK}$ and run $\mathbf{SE}.\mathbf{Enc}(M, \mathcal{SK})$ to obtain a symmetric ciphertext $\mathcal{CT}$.
  - Choose an access policy $(\mathcal{M}, \rho)$, where $\mathcal{M}$ is an $l \times n$ matrix.
  - Encrypt the symmetric key $\mathcal{SK}$ under $(\mathcal{M}, \rho)$ as follows: randomly choose $\{t_i\}_{i \in [l]} \in \mathbb{Z}_p$ and a vector $\vec{y} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. Calculate $\lambda_i = \vec{y} \cdot \mathcal{M}_i$ for $i = 1$ to l, where $\mathcal{M}_i$ is the ith row of $\mathcal{M}$. Set the ciphertext as
    $$CT = (C_1 = \mathcal{SK} \cdot e(g,g)^{\alpha s}, C_2 = g^s,$$
    $$\{D_{1,i} = g^{a\lambda_i} A_{\rho(i)}^{-t_i}, D_{2,i} = g^{t_i}\}_{i \in [l]}, (\mathcal{M}, \rho)).$$
    Note that this step operates the same as the algorithm **Encrypt**$(PK, (\mathcal{M}, \rho), M)$ of Γ.
  - Upload $(\mathcal{CT}, CT)$ to the cloud.

- Download Request Generation :
  In order to download a shared (encrypted) file from the cloud, a data user generates a download request as follows.
  - Choose a random $r \in \mathbb{Z}_p$, set $L_1' = (L_1)^r, L_2' = (L_2)^r, \{L_{3,x}' = (L_{3,x})^r\}_{\forall x \in S}$.
  - Send the download request $DReq = (\texttt{'download'}, (L_1', L_2', \{L_{3,x}'\}_{\forall x \in S}, S))$ to the cloud.

- Access Control on Download Request :
  This procedure provides the access control over download request in the sense that only authorized data users can download the shared data. For a download request $DReq = (\texttt{'download'}, (L_1', L_2', \{L_{3,x}'\}_{\forall x \in S}, S$

$))$ for a shared (encrypted) file $(\mathcal{CT}, CT)$ (where $CT = (C_1, C_2, \{D_{1,i}, D_{2,i}\}_{i\in[l]}, (\mathcal{M}, \rho))$), the access control on download request procedure consists of the following steps:

- The cloud forms a call request as $(CT, DReq)$ and sends the request to the authority [3].
- Upon receiving the call request $(CT, DReq)$ from the cloud, the authority takes C_2 from CT and L'_2 from $DReq$ as input, computes $E_1 = e((C_2)^a, L'_2) = e(g,g)^{savr}$. It then sends E_1 to the cloud.
- If S does not satisfy $(\mathcal{M}, \rho)$, the cloud sets $E_2 = \perp$. Otherwise, let $I \subset [l]$ be defined as $I = \{i : \rho(i) \in S\}$ and the cloud computes the constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} w_i\lambda_i = s$. The cloud then calculates

$$E_2 = \prod_{i\in I}(e(D_{1,i}, L'_2)e(D_{2,i}, L'_{3,\rho(i)}))^{w_i}.$$

  Note that if the data user is authorized (i.e., S satisfies the access policy $(\mathcal{M}, \rho)$ defined over CT), then E_2 equals $e(g,g)^{avsr}$.
- The cloud checks whether the following equation holds or not: $E_2 = E_1$. If yes, it sends the shared (encrypted) file $(\mathcal{CT}, CT)$ to the data user. Otherwise, it concludes that the data user is not authorized for the file, and ignores the data user's download request.

- Access Shared Data :

  In this procedure, an authorized data user decrypt a received (encrypted) file using his secret access credential to access the underlying data. Specifically, upon receiving a shared (encrypted) file $(\mathcal{CT}, CT)$ (where $CT = (C_1, C_2, \{D_{1,i}, D_{2,i}\}_{i\in[l]}, (\mathcal{M}, \rho))$), an authorized data user does the following steps to recover the shared data (i.e., plaintext).
  - Let $I \subset [l]$ be defined as $I = \{i : \rho(i) \in S\}$. Compute the constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} w_i\lambda_i = s$. Calculate

$$E = \prod_{i\in I}(e(D_{1,i}, L_2)e(D_{2,i}, L_{3,\rho(i)}))^{w_i} = e(g,g)^{avs},$$

$$F = e(C_2, L_1)/E = e(g,g)^{\alpha s}, \mathcal{SK} = C_1/F.$$

    Note that this step operates the same as the algorithm $\mathbf{Decrypt}(CT, SK)$ of Γ.
  - Run $\mathbf{SE.Dec}(\mathcal{CT}, \mathcal{SK})$ to obtain the plaintext M.

4.3 The Enhanced System

In the basic construction, the authority must be always online. It is desirable that the cloud can check the download request by itself. In this subsection, to address this issue, we present an enhanced system. The procedures Data User Registration, Shared File Generation and Outsourcing, Download Request Generation, Access Shared Data are the same as those of the basic system, the remaining algorithms are modified as follows.

- Parameter Initialization :

  This procedure is almost the same with that of the basic system, excepting for the following additional steps (that follows the last step of the basic system):
  - The cloud equipped with SGX processors creates an enclave [4].
  - The authority prepares a SGX program $\mathcal{C}$ for realizing the following functionality: Upon receiving an input h, compute $E'_1 = (h)^{s'}$ and output E'_1, where s' is the internal secret inside an enclave.
  - The authority establishes a secure channel with the enclave, and securely loads the code of program $\mathcal{C}$ and the master secret parameter a to the enclave, using for instance AES-GCM for confidentiality and integrity protection [26] (In particular, the authority uses a randomly generated secret key to encrypt the code and the data, and employs the secure channel to share the secret key with the enclave).
  - The enclave keeps a as its internal secret (i.e., sets $a = s'$).

  In order to verify the software running in the enclave on the cloud side, the authority uses remote attestation [2] to check the integrity of the code (i.e., the program $\mathcal{C}$) and static data (i.e., the master secret parameter a) loaded into the enclave [26] (please refer to Subsection 2.6 for more details about remote attestation).

- Access Control on Download Request :

  The procedure is almost the same as that of the basic system, excepting for replacing the first step with the following steps:
  - The cloud sends a call request to the enclave with C_2 (of CT) as input.
  - Upon receiving the call request with C_2, the enclave runs program $\mathcal{C}$ with C_2 as input (i.e., calculates $E'_1 = (C_2)^a$) and returns E'_1 to the cloud.
  - The cloud computes $E_1 = e(E'_1, L'_2) = e(g,g)^{savr}$.

Side-channel resilience. Although the security of SGX is evolving, it is still susceptible to a number of side-channel attacks [6], [14], [30], [37]. One defense against these side-channel attacks is to ensure that the enclave program is data-oblivious. That is, the program will not include control flow branches or memory access patterns that depend on the values of sensitive data [7], [13]. Another approach is to employ the technique of ORAM [27]. For the enhanced system, the only enclave operations that touch secret data are decryption operations (for loading the data via AES-GCM) and the specific function (that compute $E'_1 = (h)^{s'}$ and output E'_1). In our implementation of AES-GCM, we utilize the SGX SDK cryptographic library, therefore, it is resilient to software-based side-channels (which is similar to [7]). For the function, we implemented it in a way that it achieves the property of data-oblivious (i.e., control flow branches or memory access patterns will not depend on the sensitive data). Therefore, the enhanced system is secure against side channel attack.

3. To save bandwidth, the cloud could simply take C_2 from CT and L'_2 from $DReq$ to form the call request as (C_2, L'_2).

4. Note that a cloud equipped with SGX processors can create multiple enclaves if necessary.

5 Security Analysis

In this section, we present the security analyses of the two proposed systems on how they achieve the security requirements listed in Section 3.

5.1 Security of the basic system

(1) Security against honest-but-curious cloud

For simplicity, we denote by Γ_1, Γ_2 the CP-ABE scheme in [36] and the basic system in Subsection 4.2, respectively.

Lemma 1. *[36] If the decisional q-Parallel BDHE assumption holds, Γ_1 is IND-CPA secure.*

Lemma 2. *If Γ_1 is IND-CPA secure, Γ_2 is IND-CPA secure.*

Proof. To prove the security of Γ_2, we suppose there exists a PPT adversary $\mathcal{A}_2$ with a challenge access policy (M^*, ρ^*) (M^* is an $l \times n$ matrix) that has a non-negligible advantage in breaking Γ_2. We build a PPT simulator algorithm $\mathcal{A}_1$ that has a non-negligible advantage in breaking Γ_1.

- **Init**: $\mathcal{A}_1$ gets the challenge access policy (M^*, ρ^*) from $\mathcal{A}_2$ and sends the received (M^*, ρ^*) to the Γ_1 challenger.
- **Setup**: $\mathcal{A}_1$ receives the public parameters $pk = (g, g^a, h_1, ..., h_U, e(g,g)^\alpha)$ from the Γ_1 challenger. It sends pk to $\mathcal{A}_2$.
- **Phase 1**: For the secret key query from $\mathcal{A}_2$ with an attribute set S (with a restriction that S does not satisfy (M^*, ρ^*)), $\mathcal{A}_1$ sends it to the Γ_1 challenger and obtains a secret key $SK'_S = (K', L', \{K'_x\}_{x\in S})$. $\mathcal{A}_1$ then sets $L_1 = K', L_2 = L', \{L_{3,x} = K'_x\}_{x\in S})$ and returns $SK = (L_1, L_2, \{L_{3,x}\}_{x\in S}))$ to $\mathcal{A}_2$.
- **Challenge**: $\mathcal{A}_2$ declares two equal length messages (m_0, m_1) and sends them to $\mathcal{A}_1$. $\mathcal{A}_1$ chooses two random symmetric key $\mathcal{SK}_0, \mathcal{SK}_1$, sends them to the Γ_1 challenger and obtains a challenge ciphertext $CT^* = (C^*, C'^*, \{C^*_{x,1}, C^*_{x,2}\}_{x\in[l]})$. $\mathcal{A}_1$ selects a random bit $b_{\mathcal{A}_1} \in \{0,1\}$, and runs $\mathbf{SE}.\mathbf{Enc}(M, \mathcal{SK}_{b_{\mathcal{A}_1}})$ to obtain symmetric ciphertexts $\mathcal{CT}^*$. It then returns the new challenge ciphertext $\mathcal{CT}^*, CT^*$ to $\mathcal{A}_2$.
- **Query Phase 2**: Same as **Query Phase 1**.
- **Guess**: $\mathcal{A}_2$ outputs a guess $b \in \{0,1\}$ and sends it to $\mathcal{A}_1$. $\mathcal{A}_1$ sends the received b to the Γ_1 challenger.

Note that the distributions of the public parameters, challenge ciphertext and decryption keys in the above game are the same as that of the real system, if $\mathcal{A}_2$ can break Γ_2 with a non-negligible advantage, $\mathcal{A}_1$ can break Γ_1 with the same advantage. □

Lemma 3. *If the decisional q-Parallel BDHE assumption holds, the cloud cannot identify the owner of any newly uploaded file.*

Proof. From Lemma 1 and Lemma 2, we have that Γ_2 is IND-CPA secure. Hence, the shared file $(\mathcal{CT}, CT)$ does not contain any information that can be used to make inference about the owner of the file. We conlclude that the cloud cannot obtain any useful information to know the owner. □

Lemma 4. *If the decisional q-Parallel BDHE assumption holds, the cloud cannot obtain the plaintext of the encrypted data stored on it.*

Proof. According to the *Shared File Generation and Outsourcing* phase, we know that the encrypted data $(\mathcal{CT}, CT)$ is generated based on a hybrid system of the symmetric-key encryption scheme $\mathbf{SE}$ and the CP-ABE scheme Γ_1 [36]. From Lemma 1 and Lemma 2, we have that the basic system in Subsection 4.2 Γ_2 is IND-CPA secure. Due to the security property of Γ_2, it follows from the security result of hybrid encryption system [8] that the encrypted data $(\mathcal{CT}, CT)$ can only be decrypted with valid secret keys. Since the cloud cannot obtain such secret keys, it cannot decrypt the file. □

Lemma 5. *If the decisional q-Parallel BDHE assumption holds, the cloud cannot identify the sender of any download request.*

Proof. From Lemma 1 and Lemma 2, we have that the basic system in Subsection 4.2 is IND-CPA secure. Hence, for any download request $DReq = (\texttt{'download'}, (L'_1, L'_2, \{L'_{3,x}\}_{\forall x\in S}, S))$, it does not leak any information that can be used to make inference about its sender. We conclude that the cloud cannot obtain any useful information to identify the sender. □

Theorem 1. *The basic system in Subsection 4.2 is secure against honest-but-curious cloud.*

Proof. It follows directly from Lemma 3, Lemma 4, and Lemma 5. □

(2) Security against *malicious* data user

Lemma 6. *If the decisional q-Parallel BDHE assumption holds, any unauthorized data user cannot download the shared file(s).*

Proof. In order to download a shared file $(\mathcal{CT}, CT)$ (where $CT = (C_1, C_2, \{D_{1,i}, D_{2,i}\}_{i\in[l]}, (\mathcal{M}, \rho))$) from the cloud, a download request sent by any data user has to pass the check on the cloud side. Specifically, for a download request $DReq = (\texttt{'download'}, (L'_1, L'_2, \{L'_{3,x}\}_{\forall x\in S}, S))$, it passes the check on the cloud side if the following two conditions are satisfied: (1) $(\mathcal{M}, \rho)$ is satisfied by S; and (2) the equation $E_1 = E_2$ holds, where $E_1 = e((C_2)^a, L'_2)$, $E_2 = \prod_{i\in I}(e(D_{1,i}, L'_2)e(D_{2,i}, L'_{3,\rho(i)}))^{w_i}$ (as described in Section 4). Since the authority is fully trusted, we have that $E_1 = e((C_2)^a, L'_2) = e(g,g)^{savr}$. Suppose there exists an adversary (i.e., unauthorized data user) that can construct a download request such that the equation $E_2 = e(g,g)^{avsr}$ (i.e., $E_2 = E_1$) holds during the procedure Access Control on Download Request. That is, the adversary can construct a download request that satisfies the above conditions (1) and (2). It implies that the adversary can construct a download request that is derived from a valid secret key. It implies that the adversary can construct such valid secret key, which breaks the IND-CPA security of the basic system in Subsection 4.2. However, from Lemma 1 and Lemma 2, we have that the basic system in Subsection 4.2 is IND-CPA secure. □

Lemma 7. *If the decisional q-Parallel BDHE assumption holds, any unauthorized data user cannot decrypt the shared file even if the data user obtains the file.*

Proof. It follows directly from Lemma 2. □

Theorem 2. *The basic system in Subsection 4.2 is secure against malicious data user.*

Proof. It follows directly from Lemma 6 and Lemma 7. □

5.2 Security of the enhanced system

(1) Security against honest-but-curious cloud

Theorem 3. *The enhanced system in Subsection 4.2 is secure against honest-but-curious cloud.*

Proof. The proof of this theorem is the same with that of Theorem 1. □

(2) Security against *malicious* data user

Let Γ'_1, Γ'_2 be the CP-ABE scheme in [36] and the enhanced system in Subsection 4.3, respectively.

Lemma 8. *If Γ'_1 is IND-CPA secure, Γ'_2 is IND-CPA secure.*

Proof. Since the procedures Data User Registration, Shared File Generation and Outsourcing of Γ'_2 are the same as those of the basic system, the proof of this lemma is the same with that of Lemma 2. □

Lemma 9. *Any unauthorized data user cannot download the shared file(s).*

Proof. Similar to the basic system, in order to download a shared file $(\mathcal{CT}, CT)$ (where $CT = (C_1, C_2, \{D_{1,i}, D_{2,i}\}_{i\in[l]}, (\mathcal{M}, \rho))$) from the cloud, a download request sent by any data user has to pass the check on the cloud side. Specifically, for a download request $DReq = (\texttt{'download'}, (L'_1, L'_2, \{L'_{3,x}\}_{\forall x\in S}, S))$, it can pass the check on the cloud side if the following two conditions are satisfied: (1) $(\mathcal{M}, \rho)$ is satisfied by S; and (2) the equation $E_1 = E_2$ holds, where $E_1 = e((C_2)^a, L'_2)$, $E_2 = \prod_{i\in I}(e(D_{1,i}, L'_2)e(D_{2,i}, L'_{3,\rho(i)}))^{w_i}$ (as described in Section 4). Due to the isolation functionality of SGX, the code and data (i.e., $\mathcal{C}$ and a) inside the enclave protected memory cannot be modified by any process external to the enclave. Hence, E'_1 always equals $(C_2)^a$ and E_1 always equals $e(g,g)^{savr}$. Suppose there exists an unauthorized adversary that can construct a download request such that the equation $E_2 = e(g,g)^{avsr}$ (i.e., $E_2 = E_1$) holds during the procedure Access Control on Download Request, i.e., the above conditions (1) and (2) are satisfied. It implies that the adversary can construct a download request that is derived from a valid secret key. Since the adversary is unauthorized, it implies that the adversary can construct such valid secret key, which breaks the IND-CPA security of the enhanced system in Subsection 4.3. However, from Lemma 1 and Lemma 8, we have that the enhanced system in Subsection 4.3 is IND-CPA secure. □

Lemma 10. *Any unauthorized data user cannot decrypt the shared file even if the data user obtains the file.*

Proof. It follows directly from Lemma 8. □

Theorem 4. *The enhanced system in Subsection 4.3 is secure against malicious data user.*

Proof. It follows directly from Lemma 9 and Lemma 10. □

6 Performance Evaluations

6.1 Theoretical Analysis

Since the two proposed systems are built on the top of the CP-ABE system in [36], in this subsection, we first give a theoretical analysis of the comparison between the two proposed systems and the (underlying) CP-ABE system in [36]. Let Σ_0, Σ_1, Σ_2 be the CP-ABE system in [36], the basic system in subsection 4.2 and the enhanced system in subsection 4.3, respectively. Table 1 gives the comparison in terms of computational cost. In particular, the computational cost of Parameter Initialization of Σ_1 (resp. Σ_2) is the same (resp. almost the same) as the algorithm **Setup**$(\lambda, \mathcal{U})$ of Σ_0, excepting that it adds a into the master secret key MSK. Furthermore, the generation of secret key of Σ_1 (resp. Σ_2) is the same with that of Σ_0. In addition, the computational costs of encryption and decryption of Σ_1 (resp. Σ_2) are the same with that of Σ_0 [5]. That is, compared with Σ_0, the two proposed systems do not impose any additional computational cost. Table 2 gives the comparison in terms of communication cost. In particular, the public parameters size, secret key size, ciphertext size of Σ_1 (resp. Σ_2) are all the same with that of Σ_0. We note that the technique used to fulfill the feature of access control on download request is "transplantable" to other CP-ABE.

Table 3 gives the comparison among the strawman approach described in the Introduction, our proposed systems and the related work in terms of computational cost. For a fair comparison, for each computational cost in [38], we only count the computational cost which is used for access control on download request. The computational costs for procedures Parameter Initialization, Shared File Generation and Outsourcing and access control on download request on the data user side of our proposed systems are less (or much less) than that of [38]. In contrast, the access control on download request on the cloud side of our proposed systems require more computations. This exactly reflects the main design philosophy: to move expensive computations to the cloud as many as possible. Table 4 gives the comparison among the strawman approach described in the Introduction, our proposed systems and the related work in terms of communication cost. For a fair comparison, we only count the communication cost in [38] which is used for access control on download request. It shows that the communication cost for download request of our proposed systems are less than that of [38]. In particular, the ciphertext size of our proposed systems are much less than that of [38].

6.2 Experimental Analysis

To evaluate the practical performance, we implement the two proposed systems within the *Charm* framework [1], where 224-bit MNT elliptic curves from Pairing-Based Cryptography library [18] is used. The experiments are performed in test beds of two PCs. The first PC plays the roles of data owner and data user, the second PC plays the role of authority and cloud. The hardware and software of the first PC are as follows: Intel Core i7-7700M CPU @3.6 GHz,

5. For fair comparison, Σ_0 is considered in KEM/DEM setting (i.e., with the same setting as Σ_1 and Σ_2).

TABLE 1 Comparison between the two proposed systems and the CP-ABE system in [36] in terms of computational cost[1]

	PI	DUR	SFG	DRG	ACC	ASD
[36]	$P+E$	$(3+\|S\|)E$	$E_s+(1+3l)E+E_{G_T}$ [3]	$\times$	$\times$	$D_s+(1+2\|I\|)P$ [3]
P1	$P+E$	$(3+\|S\|)E$	$E_s+(1+3l)E+E_{G_T}$	$(2+\|S\|)E$	$2\|I\|P$	$D_s+(1+2\|I\|)P$
P2	$P+E$ [2]	$(3+\|S\|)E$	$E_s+(1+3l)E+E_{G_T}$	$(2+\|S\|)E$	$(1+2\|I\|)P$	$D_s+(1+2\|I\|)P$

[1] P1 stands for the basic system, P2 stands for the enhanced system, PI stands for the computational cost of the procedure Parameter Initialization (i.e., the **Setup** algorithm of the CP-ABE system in [36]), DUR stands for the computational cost of the procedure Data User Registration (i.e., the **KeyGen** algorithm of Σ_0), SFG stands for the computational cost of the procedure Shared File Generation and Outsourcing (i.e., the **Encrypt** algorithm of Σ_0), DRG stands for the computational cost of the procedure Download Request Generation, ACC stands for the computational cost of the procedure Access Control on Download Request on the cloud side, and ASD stands for the computational cost of the procedure Access Shared Data (i.e., the **Decrypt** algorithm of Σ_0). Let P be a pairing operation, E be an exponentiation in G, E_{G_T} be an exponentiation in G_T, $|S|$ be the size of the attribute set of a secret key, l be the size of an access policy, $|I|$ be the number of attributes in a secret key that satisfies a ciphertext's access policy, E_s be the computational cost for the encryption of the underlying symmetric-key encryption scheme, and D_s be the computational cost for the decryption of the underlying symmetric-key encryption scheme.

[2] The maximum amounts of time to load the SGX program $\mathcal{C}$ and the master secret parameter a to the enclave is much smaller than the maximum amounts of time to compute an exponentiation in G or a pairing, hence, we ignore it here.

[3] For fair comparison, the CP-ABE system in [36] is considered in the KEM/DEM setting.

TABLE 2 Comparison between the two proposed systems and the CP-ABE system in [36] in terms of communication cost[1]

	PKS	MSKS	SKS	DRS	CS [2]
[36]	$\approx(1.215+0.405\|\mathcal{U}\|)$ KB	≈ 0.405 KB	$\approx(0.81+0.405\|S\|)$ KB	$\times$	$\approx(0.81+0.81l)$ KB
P1	$\approx(1.215+0.405\|\mathcal{U}\|)$ KB	≈ 0.469 KB	$\approx(0.81+0.405\|S\|)$ KB	$\approx(0.81+0.405\|S\|)$ KB	$\approx(0.81+0.81l)$ KB
P2	$\approx(1.215+0.405\|\mathcal{U}\|)$ KB	≈ 0.469 KB	$\approx(0.81+0.405\|S\|)$ KB	$\approx(0.81+0.405\|S\|)$ KB	$\approx(0.81+0.81l)$ KB

[1] P1 stands for the basic system, P2 stands for the enhanced system, PKS stands for the size of the public parameters, MSKS stands for the size of the master secret key, SKS stands for the size of secret key, DRS stands for the size of download request, and CS stands for the size of ciphertext (the CP-ABE part, i.e., CT in Subsection 4.2). Let $\mathcal{U}$ be the attribute universe, $|S|$ be the size of the attribute set of a secret key, l be the size of an access policy.

[2] For simplicity, we omit the symmetric part of the ciphertext (i.e., the symmetric ciphertext $\mathcal{CT}$ in Subsection 4.2).

TABLE 3 Comparison between the two proposed systems and other related work in terms of computational cost[1]

	PI	DUR	SFG	DRU	DRC	ASD
[38]	CPK_{ABE}+CPK_{Sig}	CSK_{ABE}	T·(CCT_{ABE}+(N+1)·CCT_s+C_{sig}+N·H)	$(1+2\|I\|)P$+CS_{dec}	H	$D_{ABE}+D_s$
P3	CPK_{ABE}	CSK_{ABE}	(N_0+1)(CCT_{ABE}+CCT_s)	$(1+2\|I\|)P$	E	$D_{ABE}+D_s$
P1	CPK_{ABE}	CSK_{ABE}	CCT_{ABE}+CCT_s	$(2+\|S\|)E$	$2\|I\|P$	$D_{ABE}+D_s$
P2	CPK_{ABE}	CSK_{ABE}	CCT_{ABE}+CCT_s	$(2+\|S\|)E$	$(1+2\|I\|)P$	$D_{ABE}+D_s$

[1] P1 stands for the basic system, P2 stands for the enhanced system, P3 stands for the strawman approach described in the Introduction (with N_0 challenge ciphertexts), PI stands for the computational cost of the procedure Parameter Initialization, DUR stands for the computational cost of the procedure Data User Registration, SFG stands for the computational cost of the procedure Shared File Generation and Outsourcing, DRU stands for the computational cost for access control on download request on the data user side, DRC stands for the computational cost for access control on download request on the cloud side, and ASD stands for the computational cost of the procedure Access Shared Data. Let CPK_{ABE} be the computational cost for generating the public parameters of the underlying CP-ABE scheme, CPK_{Sig} be the computational cost for generating the public parameters of the signature scheme in [38], CSK_{ABE} be the computational cost for generating the secret key of the underlying CP-ABE scheme, CCT_{ABE} be the computational cost for generating the ciphertext of the underlying CP-ABE scheme, CCT_s be the computational cost for generating the ciphertext of the underlying symmetric-key encryption scheme, C_{sig} be the computational cost for generating the signature of the signature scheme in [38], H be the computational cost of the hash function in [38], T be the number of update for challenges in [38], N be the number of challenge plaintexts in [38], $|I|$ be the number of attributes in a secret key that satisfies a ciphertext's access policy, P be a pairing operation, CS_{dec} be the computational cost for decryption of the underlying symmetric-key encryption scheme, E be an exponentiation in G, $|S|$ be the size of the attribute set of a secret key, D_{ABE} be the computational cost for the decryption of the underlying CP-ABE scheme, and D_s be the computational cost for the decryption of the underlying symmetric-key encryption scheme.

TABLE 4 Comparison between the two proposed systems and other related work in terms of communication cost[1]

	PKS	SKS	DRS	CS
[38]	SPK_{ABE}+SPK_{sig}	SSK_{ABE}	SCT_{ABE}+SCT_s+SP_s	T·(SCT_{ABE}+(N+1)·SCT_s+SC_{sig}+N·S_H)
P3	SPK_{ABE}	SSK_{ABE}	SCT_{ABE}+SCT_s	$(N_0+1)\mathrm{SCT}_{ABE}$
P1	SPK_{ABE}	SSK_{ABE}	SSK_{ABE}	SCT_{ABE}
P2	SPK_{ABE}	SSK_{ABE}	SSK_{ABE}	SCT_{ABE}

[1] P1 stands for the basic system, P2 stands for the enhanced system, P3 stands for the strawman approach described in the Introduction (with N_0 challenge ciphertexts), PKS stands for the size of the public parameters, SKS stands for the size of secret key, DRS stands for the communication cost for download request, and CS stands for the size of ciphertext. Let SPK_{ABE} be the size of public parameters of the underlying CP-ABE scheme, SPK_{sig} be the size of the public parameters of the signature scheme in [38], SSK_{ABE} be the size of secret key of the underlying CP-ABE scheme, SCT_{ABE} be the ciphertext size of the underlying CP-ABE scheme, SCT_s be the ciphertext size of the underlying symmetric-key encryption scheme, SP_s be the plaintext size of the underlying symmetric-key encryption scheme, SC_{sig} be the signature size of the signature scheme in [38], S_H be the size of a hash in [38], T be the number of update for challenges in [38], N be the number of challenge plaintexts in [38].

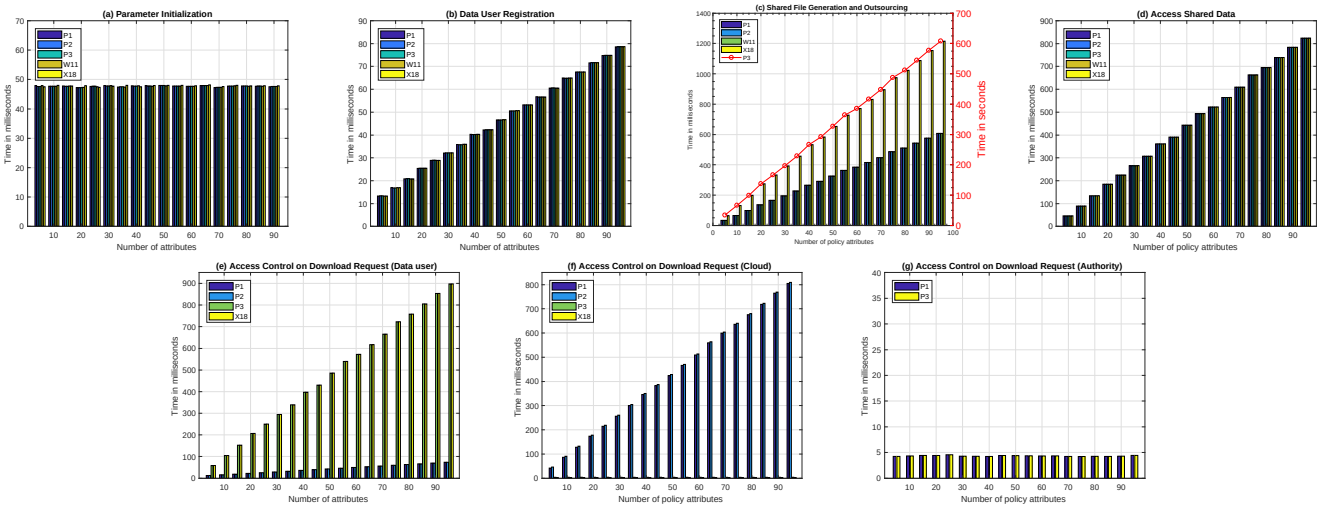


Fig. 2 Experimental results in terms of computational cost

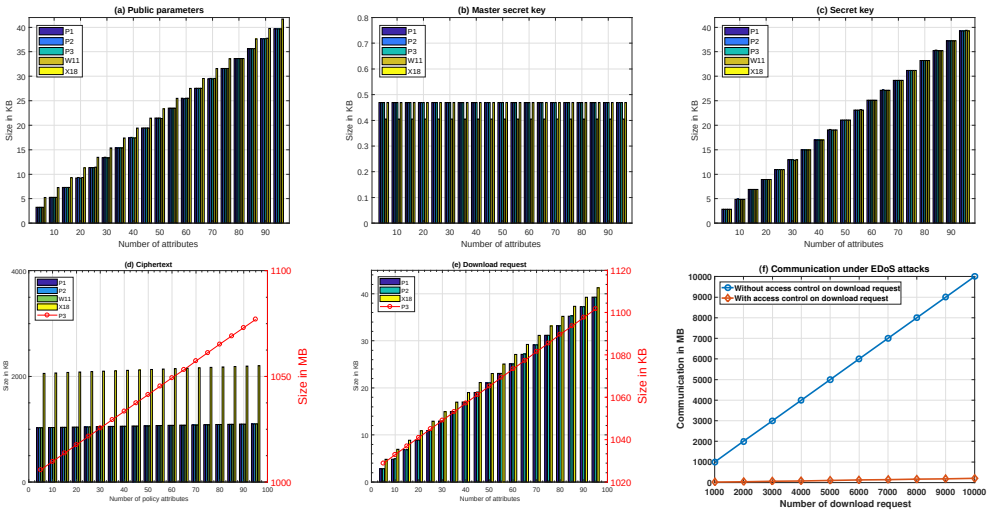


Fig. 3 Experimental results in terms of communication cost

32 GB RAM, Ubuntu 16.04 LTS 64-bit, Python 2.7.12. The hardware and software of the second PC are as follows: Intel Core i7-7700 CPU @3.6 GHz, 32 GB RAM, Ubuntu 16.04 LTS 64-bit, Intel(R) SGX SDK 2.1.102.43402 and Intel(R) SGX PSW 2.1.102.43402, Python 2.7.12 and C/C++. In order to simulate the worst case, we generate ciphertext policies (of CP-ABE) in the form of (A_1 and A_2 ... and A_l), where A_i is an attribute. We set 20 distinct access policies in this form with the number of policy attributes increasing from 5 to 95, repeat each instance 20 times and take the average. All the instances are kept completely independent to each other. The time is given in milliseconds.

We aim to evaluate the efficiency of the basic system (denoted by P1) and the enhanced system (denoted by P2) by comparing the running time taken in each procedure with the underlying CP-ABE in [36] (denoted by W11, which does not support the functionality of access control over download request), the strawman approach described in the Introduction (denoted by P3, where the number of challenger ciphertext is 1000) and the related approach proposed in [38] (denoted by X18, where the number of update for challenges is set to be 2, the number of challenge plaintexts is 1000). Fig. 2 shows the experimental results in terms of computational cost. In particular, Fig. 2(a), 2(b), 2(c) and 2(d) illustrate that the time costs for procedures Parameter Initialization, Data User Registration, Shared File Generation and Outsourcing, and Access Shared Data of the basic system and the enhanced system are the same (or almost the same) with that of the underlying CP-ABE in [36]. Fig. 2(c) shows that the computational costs for shared files of our proposed systems are less than that of the strawman approach and the approach in [38]. Fig. 2(e) shows that the time cost for the procedure Download Request Generation of the basic system is the same with that of the enhanced system, and less than that of the strawman approach and the approach in [38]. Fig. 2(f) illustrates that the time cost executed on the cloud side for the procedure Access Control on Download Request of the enhanced system is a little higher than that of the basic system. Fig. 2(g) shows the time cost executed on the authority side for the procedure Access Control on Download Request of the basic system, which only costs 4.36 ms on

average. Fig. 3 shows the experimental results in terms of communication cost. In particular, Fig. 3(a), 3(c), 3(d) illustrate that the communication costs for procedures Parameter Initialization, Data User Registration, Shared File Generation and Outsourcing of the basic system and the enhanced system are the same with that of the underlying CP-ABE in [36]. Fig. 3(a) shows that the communication cost for procedure Parameter Initialization of our proposed systems are less than that of the approach in [38]. Fig. 3(d) illustrates that the communication costs for ciphertexts of our proposed systems are less than that of the strawman approach and the approach in [38]. Fig. 3(e) shows that the communication cost for the procedure Download Request Generation of the basic system is the same with that of the enhanced system, and less than that of the strawman approach and the approach in [38]. Fig. 3(f) shows that our proposed systems dramatically reduce the communication overhead under EDoS attacks, where the number of attribute for each download request is 50 and the file size is 1MB. To sum up, our proposed systems achieve the feature of access control on download request without incurring significant overhead compared to the underlying CP-ABE in [36].

7 Conclusion and Future Work

We addressed an interesting and long-lasting problem in cloud-based data sharing, and presented two dual access control systems. The proposed systems are resistant to DDoS/EDoS attacks. We state that the technique used to achieve the feature of control on download request is "transplantable" to other CP-ABE constructions. Our experimental results show that the proposed systems do not impose any significant computational and communication overhead (compared to its underlying CP-ABE building block).

In our enhanced system, we employ the fact that the secret information loaded into the enclave cannot be extracted. However, recent work shows that enclave may leak some amounts of its secret(s) to a malicious host through the memory access patterns [37] or other related side-channel attacks [14], [30]. The model of *transparent enclave execution* is hence introduced in [35]. Constructing a dual access control system for cloud data sharing from transparent enclave is an interesting problem. In our future work, we will consider the corresponding solution to the problem.

References

[1] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[2] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for cpu based attestation and sealing. In *Workshop on hardware and architectural support for security and privacy (HASP)*, volume 13, page 7. ACM New York, NY, USA, 2013.

[3] Alexandros Bakas and Antonis Michalas. Modern family: A revocable hybrid encryption scheme based on attribute-based encryption, symmetric searchable encryption and SGX. In *SecureComm 2019*, pages 472–486, 2019.

[4] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[5] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *S&P 2007*, pages 321–334. IEEE, 2007.

[6] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptology ePrint Archive*, 2016(086):1–118, 2016.

[7] Ben Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. IRON: functional encryption using intel SGX. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, pages 765–782, 2017.

[8] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology-CRYPTO 1999*, pages 537–554. Springer, 1999.

[9] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, pages 89–98. ACM, 2006.

[10] Jinguang Han, Willy Susilo, Yi Mu, Jianying Zhou, and Man Ho Allen Au. Improving privacy and security in decentralized ciphertext-policy attribute-based encryption. *IEEE transactions on information forensics and security*, 10(3):665–678, 2015.

[11] Christofer Hoff. Cloud computing security: From ddos (distributed denial of service) to edos (economic denial of sustainability). *http://www. rationalsurvivability.com/blog/?p=66*.

[12] Joseph Idziorek, Mark Tannian, and Doug Jacobson. Attribution of fraudulent resource consumption in the cloud. In *IEEE CLOUD 2012*, pages 99–106. IEEE, 2012.

[13] Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel® software guard extensions: Epid provisioning and attestation services. *White Paper*, 1:1–10, 2016.

[14] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside sgx enclaves with branch shadowing. In *26th USENIX Security Symposium, USENIX Security*, pages 16–18, 2017.

[15] Jiguo Li, Xiaonan Lin, Yichen Zhang, and Jinguang Han. Ksf-oabe: outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Transactions on Services Computing*, 10(5):715–725, 2017.

[16] Jiguo Li, Yao Wang, Yichen Zhang, and Jinguang Han. Full verifiability for outsourced decryption in attribute based encryption. *IEEE Transactions on Services Computing, DOI: 10.1109/TSC.2017.2710190*, 2017.

[17] Wei Li, Kaiping Xue, Yingjie Xue, and Jianan Hong. Tmacs: A robust and verifiable threshold multi-authority access control system in public cloud storage. *IEEE Transactions on parallel and distributed systems*, 27(5):1484–1496, 2016.

[18] Ben Lynn et al. The pairing-based cryptography library. *Internet: crypto. stanford. edu/pbc/[Mar. 27, 2013]*, 2006.

[19] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative instructions and software model for isolated execution. In *HASP@ISCA 2013*, page 10, 2013.

[20] Antonis Michalas. The lord of the shares: combining attribute-based encryption and searchable encryption for flexible data sharing. In *SAC 2019*, pages 146–155, 2019.

[21] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Kaitai Liang, Hui Ma, and Lifei Wei. Auditable σ-time outsourced attribute-based encryption for access control in cloud computing. *IEEE Transactions on Information Forensics and Security*, 13(1):94–105, 2018.

[22] Jianting Ning, Zhenfu Cao, Xiaolei Dong, and Lifei Wei. White-box traceable CP-ABE for cloud storage service: How to catch people leaking their access credentials effectively. *IEEE Transactions on Dependable and Secure Computing*, 15(5):883–897, 2018.

[23] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Lifei Wei, and Xiaodong Lin. Large universe ciphertext-policy attribute-based encryption with white-box traceability. In *Computer Security-ESORICS 2014*, pages 55–72. Springer, 2014.

[24] Jianting Ning, Xiaolei Dong, Zhenfu Cao, and Lifei Wei. Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In *Computer Security–ESORICS 2015*, pages 270–289. Springer, 2015.

[25] Jianting Ning, Xiaolei Dong, Zhenfu Cao, Lifei Wei, and Xiaodong Lin. White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes. *IEEE Transactions on Information Forensics and Security*, 10(6):1274–1288, 2015.

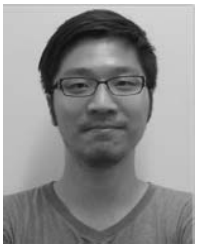
[26] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *USENIX Security Symposium*, pages 619–636, 2016.

[27] Ashay Rane, Calvin Lin, and Mohit Tiwari. Raccoon: Closing digital side-channels through obfuscated execution. In *24th USENIX Security Symposium, USENIX Security 2015*, pages 431–446, 2015.
[28] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 98–107. ACM, 2002.
[29] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.
[30] Ming-Wei Shih, Sangho Lee, Taesoo Kim, and Marcus Peinado. T-sgx: Eradicating controlled-channel attacks against enclave programs. In *NDSS 2017*, 2017.
[31] Victor Shoup. A proposal for an iso standard for public key encryption (version 2.1). *IACR Eprint Archive*, 112, 2001.
[32] Gaurav Somani, Manoj Singh Gaur, and Dheeraj Sanghi. D-dos/edos attack in cloud: affecting everyone out there! In *SIN 2015*, pages 169–176. ACM, 2015.
[33] Mohammed H Sqalli, Fahd Al-Haidari, and Khaled Salah. Edos-shield-a two-steps mitigation technique against edos attacks in cloud computing. In *UCC 2011*, pages 49–56. IEEE, 2011.
[34] Willy Susilo, Peng Jiang, Fuchun Guo, Guomin Yang, Yong Yu, and Yi Mu. Eacsip: Extendable access control system with integrity protection for enhancing collaboration in the cloud. *IEEE Transactions on Information Forensics and Security*, 12(12):3110–3122, 2017.
[35] Florian Tramer, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In *EuroS&P 2017*, pages 19–34. IEEE, 2017.
[36] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.
[37] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *S&P 2015*, pages 640–656. IEEE, 2015.
[38] Kaiping Xue, Weikeng Chen, Wei Li, Jianan Hong, and Peilin Hong. Combining data owner-side and cloud-side access control for encrypted cloud storage. *IEEE Transactions on Information Forensics and Security*, 2018.
[39] Shui Yu, Yonghong Tian, Song Guo, and Dapeng Oliver Wu. Can we beat ddos attacks in clouds? *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2245–2254, 2014.

Jianting Ning received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2016. He is currently a research fellow at School of Information Systems, Singapore Management University and a Professor with Fujian Normal University, China. His research interests include applied cryptography and information security. He has published papers in major conferences/journals such as ACM CCS, ESORICS, IEEE TIFS, IEEE TDSC, etc.

Xinyi Huang received his Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security. He has published papers in major conferences/journals such as ACM CCS, IEEE TIFS, IEEE TDSC, etc.

Willy Susilo received the bachelors degree (Summa Cum Laude) predicate in computer science from Universitas Surabaya, Indonesia, and the masters and D.Phil. degrees from UOW. He is the Director of the Institute of Cybersecurity and Cryptology. His main research interests include cryptography and cyber security. He received the prestigious ARC Future Fellowship from the Australian Research Council. He also received the UOW Researcher of the Year 2016 due to his research excellence.



Kaitai Liang (M15) received the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2014. He is currently an Assistant Professor with the Department of Computer Science, University of Surrey, Guildford, U.K. His current research interests include applied cryptography and information security in particular, encryption, blockchain, post-quantum crypto, privacy enhancing technology, and security in cloud computing.

Ximeng Liu (S13-M16) received Ph.D. degrees in Cryptography from Xidian University, China, in 2015. Now, he is a professor at College of Mathematics and Computer Science, Fuzhou University, China. He has published over 100 research articles include IEEE TIFS, IEEE TDSC, IEEE TC, IEEE TII IEEE TSC and IEEE TCC. His research interests include cloud security, applied cryptography and big data security.

Yinghui Zhang (M'18) received his Ph.D degree in Cryptography from Xidian University, China, in 2013. He is a professor at School of Cyberspace Security, National Engineering Laboratory for Wireless Security (NELWS), Xi'an University of Posts & Telecommunications. He has published research articles in ACM ASIACCS, ACM Computing Surveys, IEEE TDSC, IEEE TSC, IEEE TCC, Computer Networks, Computers & Security, etc. He served for the program committee of several conferences and the editorial members of several international journals in information security. His research interests include public key cryptography, cloud security and wireless network security.