Combining Data Owner-Side and Cloud-Side Access Control for Encrypted Cloud Storage

Kaiping Xue, *Senior Member, IEEE,* Weikeng Chen, Wei Li, Jianan Hong, and Peilin Hong

Abstract— **People endorse the great power of cloud computing, but cannot fully trust the cloud providers to host privacy-sensitive data, due to the absence of user-to-cloud controllability. To ensure confidentiality, data owners outsource encrypted data instead of plaintexts. To share the encrypted files with other users, ciphertext-policy attribute-based encryption (CP-ABE) can be utilized to conduct fine-grained and owner-centric access control. But this does not sufficiently become secure against other attacks. Many previous schemes did not grant the cloud provider the capability to verify whether a downloader can decrypt. Therefore, these files should be available to everyone accessible to the cloud storage. A malicious attacker can download thousands of files to launch economic denial of sustainability (EDoS) attacks, which will largely consume the cloud resource. The payer of the cloud service bears the expense. Besides, the cloud provider serves both as the accountant and the payee of resource consumption fee, lacking the transparency to data owners. These concerns should be resolved in real-world public cloud storage. In this paper, we propose a solution to secure encrypted cloud storages from EDoS attacks and provide resource consumption accountability. It uses CP-ABE schemes in a black-box manner and complies with arbitrary access policy of the CP-ABE. We present two protocols for different settings, followed by performance and security analysis.**

Index Terms— **Ciphertext-policy attribute-based encryption (CP-ABE), access control, public cloud storage, accounting, privacy-preserving.**

I. Introduction

CLOUD storage has many benefits, such as always-online, pay-as-you-go, and cheap [1]. During these years, more data are outsourced to public cloud for persistent storage, including personal and business documents. It brings a security concern to data owners [2]–[4]: the public cloud is not trusted, and the outsourced data should not be leaked to the cloud provider without the permission from data owners.

Many storage systems use server-dominated access control, like password-based [5] and certificate-based authentication [6]. They overly trust the cloud provider to protect their sensitive data. The cloud providers and their employees can read any document regardless of data owners' access policy. Besides, the cloud provider can exaggerate the resource consumption of the file storage and charge the payers more without providing verifiable records [2], [7], [8], since we lack a system for verifiable computation of the resource usage. Relying on the existing server-dominated access control is not secure. Data owners who store files on cloud servers still want to control the access on their own hands and keep the data confidential against the cloud provider and malicious users.

Encryption is Not Sufficient: To add the confidentiality guarantee, data owners can encrypt the files and set an access policy so that only qualified users can decrypt the document. With Ciphertext-Policy Attribute-based Encryption (CP-ABE) [9], [10], we can have both fine-grained access control and strong confidentiality [11]–[16]. However, this access control is only available for data owners, which turns out to be insufficient. If the cloud provider cannot authenticate users before downloading, like in many existing CP-ABE cloud storage systems [14], [15], the cloud has to allow everyone to download to ensure availability. This makes the storage system vulnerable to the resource-exhaustion attacks. If we resolve this problem by having data owners authenticate the downloaders before allowing them to download, we lose the flexibility of access control from CP-ABE. Here lists the two problems should be addressed in our work:

Problem I (Resource-Exhaustion Attack): If the cloud cannot do *cloud-side access control*, it has to allow anyone, including malicious attackers, to freely download, although only some users can decrypt. The server is vulnerable to resource-exhaustion attacks. When malicious users launch the DoS/DDoS attacks to the cloud storage, the resource consumption will increase. Payers (in pay-as-you-go model) have to pay for the increased consumption contributed by those attacks, which is a considerable and unreasonable financial burden. The attack has been introduced as Economic Denial of Sustainability (EDoS) [17]–[20], which means payers are financially attacked eventually. In addition, even files are encrypted,

Manuscript received July 12, 2017; revised December 15, 2017 and January 25, 2018; accepted January 31, 2018. Date of publication February 27, 2018; date of current version April 16, 2018. This work was supported in part by the National Key Research and Development Plan of China under Grant 2017YFB0801702, in part by the National Natural Science Foundation of China under Grant 61379129, in part by the Youth Innovation Promotion Association CAS under Grant 2016394, and in part by the Fundamental Research Funds for the Central Universities. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Stefan Katzenbeisser. *(Corresponding author: Kaiping Xue.)*

K. Xue is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China, and also with the Department of Information Security, University of Science and Technology of China, Hefei 230027, China (e-mail: kpxue@ustc.edu.cn).

W. Chen is with the Department of Information Security, University of Science and Technology of China, Hefei 230027, China.

W. Li, J. Hong, and P. Hong are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2018.2809679

unauthorized downloads can reduce security by bringing convenience to to offline analysis and leaking information like file length or update frequency.

Problem II (Resource Consumption Accountability): In the pay-as-you-go model, users pay money to the cloud provider for storage services. The fee is decided by resource usage. However, CP-ABE based schemes for cloud storage access control does not make online confirmations to the data owner before downloads. It is needed for the cloud service provider to prove to the payers about the actual resource usage. Otherwise, the cloud provider can charge more without being discovered [21], [22].

A. Summary of Challenges and Approaches

Challenge I (Modeling the Cloud Provider): Many existing CP-ABE based schemes [11], [12], [23] model the cloud providers (like Google, Amazon, Microsoft Azure) as semi-honest adversaries or passive attackers. However, such a definition is restricted and it excludes some possible attacks in the real world, such as exaggerated resource usage. To model such attacks, we consider a less restricted security model, covert adversary, for the cloud provider [24].

In practice, the cloud services are usually provided by some big IT enterprises like Google, Amazon, Microsoft. They need to maintain good reputation and promise secure cloud storage services to their customers. If any attempt the cloud provider *deviates from the protocol* is supposed to be caught with a possibility (e.g. $p = 0.001$), the cloud provider dares not to cheat [24], [25]. Because being caught will not only violate the service contracts, but also lead to media exposure and destroys the reputation. Aware of the aftermath, the cloud provider has to refrain from attacking, as the cheating can be detected. This model, covert security, has been used in many secure systems [26], [27].

Note that the covert security model is different with the semi-honest model. The semi-honest model, which is widely used in proxies and cloud providers, is a model that resides between "malicious" and "trusted". It models a party that observes all data, but it never executes the wrong program. Such a party will not cheat by definition, even if no other parties can detect its cheating. The covert model, which resides between "malicious" and "semi-honest", models this party differently. It will not execute the wrong program only if there is a mechanism to detect its cheating. If no detection exists in the system, the party may even compromise the data, for example. Therefore, it is more practical for public cloud storage.

Approach: model cloud providers as covert adversaries, and design protocols resilient to a covert adversary.

Challenge II (Compatible With Existing Systems): There are many constructions and variants for CP-ABE [23], [28], [29]. We don't design a new variant of CP-ABE to resolve the first challenge, as it is hard to achieve all the functionalities in these systems and also it's not necessary.

Besides the functionalities, some variants provide additional security and privacy guarantee. For example, the literatures [12], [16] hides the access policy. If the cloud-side access control makes the cloud provider knowing the access policy, it is not considered secure and compatible. It requires the cloud-side access control to be zero-knowledge for arbitrary CP-ABE schemes.

Approach: use CP-ABE in a syntactical and black-box way and ensure the construction not leaking policy and attributes. The system only learns whether the user is legitimate or not, and nothing else.

Challenge III (Minimal Performance Overhead): To protect the cloud storage effectively against the resource-exhaustion attack, the cloud-side access control needs to be efficient and lightweight, otherwise if the cloud server spends, for example 20ms, executing the cloud-side access control, it will become a computational resource exhaustion attacks, which can be used by malicious attackers for DDoS and EDoS.

The performance overhead being small also benefits the data users who download the files from the cloud storage, making the computation not approachable to resource-limited devices.

Approach: design an efficient access control for the cloud provider which should not add too much overhead.

B. Our Work and Contribution

In this paper, we combine the cloud-side access control and the existing data owner-side CP-ABE based access control, to resolve the aforementioned security problems in privacy-preserving cloud storage. Our method can prevent the EDoS attacks by providing the cloud server with the ability to check whether the user is authorized in CP-ABE based scheme, without leaking other information.

For our cloud-side access control, we use CP-ABE encryption/decryption game as challenge-response. While upload an encrypted file, the data owner firstly generates some random challenge plaintexts and the corresponding ciphertexts. The ciphertexts are related to the same access policy with the specific file. For an incoming data user, the cloud server asks him/her to decrypt randomly selected challenge ciphertext. If the user shows a correct result, which means he/she is authorized in CP-ABE, the cloud-side access control allows the file download.

To make our solution secure and efficient in real world applications, we provide two protocols of cloud-side and data owner-side combined access control. The main contribution of this work can be summarized as follows.

1) We propose a general solution to secure encrypted cloud storage to prevent the EDoS attacks, as well as have fine-grained access control and resource consumption accountability. To the best of our knowledge, this is the first work to claim that insufficient cloud-side access control in encrypted cloud storage will lead to EDoS attacks and provides a practical solution. The solution can be compatible with many CP-ABE schemes.
2) For different data owner online patterns and performance concern, we provide two protocols for authentication and resource consumption accounting. We also introduce the bloom filter and the probabilistic check to improve the efficiency but still guarantee the security.
3) Compared with many state-of-arts constructions of encrypted cloud storage that assume the existence of a semi-honest cloud provider, we use a more practical

threat model where we assume the cloud provider to be a covert adversary [24], which provides higher security guarantee.

C. Organization of the Paper

The rest of the paper is organized as follows. In Section II, we briefly review the related works. In Section III, we give some preliminaries for this paper. In Section IV, we present our system model and security model. In Section V, we give our proposed scheme in details. In Section VI and VII, we have the security and performance analysis to illustrate the correctness, efficiency and security. Finally, we conclude the paper in Section VIII.

II. Related Works

To conduct a fine-grained data owner-side access control in public cloud storage, which is semi-honest, Attribute-based Encryption(ABE) [9], [30], [31] is introduced [23]. Among ABE schemes, CP-ABE [9], [10] is practical in public cloud storage, in which the ciphertext is encrypted under an access policy and only users whose attributes satisfy the access policy can decrypt the ciphertext. Subsequently, many variants and relevant protocols [14], [16], [32], [33] have been proposed to make CP-ABE more suitable for real scenarios with rich functionalities and security properties in public cloud storage.

The cryptography-driven access control does not protect the cloud provider against many other attacks. Since the cloud provider does not conduct the access control, it cannot stop those unauthorized users. One attack that is originated from this limitation is Distributed Denial of Services (DDoS). The power of DDoS attacks has been showed to incur significant resource consumption in CPU, memory, I/O, and network [34]. The attacks can exist in public clouds [17], [35]–[37]. In [35], the limitation of cloud-side static resource allocation model is analyzed, including the risk of Economic Denial of Sustainability (EDoS) attacks, which is the case of DDoS attacks in the cloud setting in [37], or the Fraudulent Resource Consumption (FRC) attack in [17]. These attacks are intended to break the budget of public cloud customers. Some existing works try to mitigate EDoS attacks [19], [38]. In [19], the authors proposed a mitigation technique by verifying whether a request comes from a cloud user or is generated by bots. In [38], the authors proposed an attribute-based way to identify malicious clients. They treat the underlying application in a black box and do not fully immunize the attack in the algorithmic and protocol level.

Some existing works discuss the necessary of accounting resource consumption in the public cloud arouses some concerns [7], [8], [21], [22]. In the literature [21], the authors discussed key issues and challenges about how to achieve accountability in cloud computing. In the literature [22], the authors surveyed existing accounting and accountability in content distribution architectures. In the literatures [7] and [8], the authors respectively proposed a systematic approach for verifiable resource accounting in cloud computing. However, the accounting approach involves changes to the system model, and requires the anonymous verification of users, which is not supported in previous systems. Compared with relevant schemes, our approach works on the protocol level to provide the resource verifiability that relies on authorized users who satisfy the CP-ABE policy, and achieves the covert security which is more practical and secure.

III. Preliminaries

In this section, we provide the preliminary information on the cryptographic tools, the underlying CP-ABE (in Section III-A) and the encryption with integrity guarantee AEAD (in Section III-B). We present the key encapsulated mechanism to avoid double cost and ensure the integrity between the data owner and data users (in Section III-D), and we introduce the probabilistic check tool, bloom filter (in Section III-E).

The philosophy and informal definition of building a system against covert adversaries, which are stronger than many previous works, are specifically introduced here (in Section III-F).

A. CP-ABE: Ciphertext-Policy Attribute-Based-Encryption

CP-ABE is a public key encryption scheme with fine-grained access control. In CP-ABE, each user has some attributes and data owners encrypt their files with an access policy over attributes. Users in the system hold their own secret keys associated with their attribute sets. If and only if the user satisfies the access policy, the user can decrypt. Some useful definitions in CP-ABE are as follows:

Attributes: Attributes depict the party's properties relevant to access control. For example, students in EE at Berkeley may have attribute set $\{\mathsf{EE}, \mathsf{Berkeley}\}$ and students in CS at USTC may have attribute set $\{\mathsf{CS}, \mathsf{USTC}\}$.

Policy: A policy is a predicate over the attributes. For example, the policy $(\mathsf{EE} \vee \mathsf{CS})$ allows those students above to access, but none of them satisfy the policy $(\mathsf{CS} \wedge \mathsf{Berkeley})$.

Syntax: CP-ABE for security parameter $\lambda \in \mathbb{N}$ and messages $m \in \{0,1\}^{l(\lambda)}$ consists of PPT (probabilistic polynomial time) algorithms $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}\left(1^{\lambda}\right)$ generates a master public key mpk and a master key msk.
- $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathcal{A}_i)$. It takes the master secret key msk and the user's attribute set $\mathcal{A}_i$ as the input and generates a secret key sk_i associated with the attribute set $\mathcal{A}_i$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{m}, \mathbb{A})$. It takes the master public key mpk, the message m, and the access policy $\mathbb{A}$ as the input. It outputs the ciphertext ct.
- $\mathsf{m} = \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$. It takes the ciphertext ct (encrypted with access policy $\mathbb{A}$) and the secret key sk_i as input. If the attribute set $\mathcal{A}_i$ satisfies the access policy $\mathbb{A}$, it outputs the message m. Otherwise, outputs $\perp$.

Correctness, Security and the Construction: The definitions and formal proofs of correctness and security, and the construction of CP-ABE can be found in [9] and [10]. CP-ABE achieves indistinguishability under chosen-plaintext attacks.

B. Authenticated Encryption With Associated Data

Authenticated Encryption with Associated Data (AEAD) is a symmetric-key encryption that provides both confidentiality and integrity, for example, AES-GCM [39]. Here gives the syntax:

Syntax: The symmetric-key encryption AEAD for the key $\mathsf{k} \in \{0,1\}^{\lambda}$ and any message $\mathsf{m} \in \{0,1\}^{n(\lambda)}$, where $n(\lambda)$ is a polynomial-bounded function, consists of two PPT algorithms $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{c} \leftarrow \mathsf{Enc}(\mathsf{k}, \mathsf{m})$ outputs the ciphertext c for message m under key k.
- $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{k}, \mathsf{c})$ recovers the plaintext m from c under key k, but outputs $\perp$ if the result is invalid.

Furthermore, in this paper, we use the property of existential unforgeability of AEAD as:

Definition 1 (Existential Unforgeability [40]): For all security parameter $\lambda \in \mathbb{N}$, for any PPT algorithm $\mathcal{A}$:

$$\Pr\left[\mathsf{Dec}(\mathsf{k}, \mathsf{c}) \neq \perp \middle| \begin{array}{l} \mathsf{k} \leftarrow_{\$} \{0,1\}^{\lambda} \\ c \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{k}, \cdot)} \end{array}\right]$$

is negligible to λ, where $\mathcal{A}$ has never received ciphertext c from the encryption oracle $\mathsf{Enc}(\mathsf{k}, \cdot)$.

C. Digital Signature

The system uses a public-key signature scheme for message integrity. Assumed the secure distribution of public keys, any data recipient can verify the message integrity. For succinctness of signatures, we can use ECDSA:

Syntax SIG for the security parameter $\lambda \in \mathbb{N}$ and any arbitrary message $\mathsf{m} \in \{0,1\}^{n(\lambda)}$ where $n(\lambda)$ is a polynomial-bounded function, consists of three PPT algorithms $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

- $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^{\lambda})$ outputs a signing key sk_i and the corresponding verifying key vk_i.
- $\mathsf{s} \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \mathsf{m})$ generates a digital signature for the message m.
- $\mathsf{b} \in \{0,1\} \leftarrow \mathsf{Verify}(\mathsf{vk}_i, \mathsf{s}, \mathsf{m})$ outputs whether s is a valid signature of message m.

The digital signature scheme satisfies the existential unforgeability of signatures, defined as follows:

Definition 2 (Existential Unforgeability of Signatures): For every security parameter $\lambda \in \mathbb{N}$, for any PPT algorithm $\mathcal{A}$:

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{s}^*, \mathsf{m}^*) \neq 0 \middle| \begin{array}{l} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^{\lambda}) \\ (\mathsf{s}^*, \mathsf{m}^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^{\lambda}) \end{array}\right]$$

is negligible to λ, where $\mathcal{A}$ never asks for the signature for m^* from the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

D. Hybrid Encryption for CP-ABE

We illustrate the usage of hybrid encryption with the example of two CP-ABE ciphertexts with the same access policy and from the same data owner (the public key is vk_o). The cost can be reduced by encrypting an ephemeral key for both ciphertexts, described as follows:

$$\begin{pmatrix} \mathsf{ct}_0 \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{m}_0, \mathbb{A}) \\ \mathsf{ct}_1 \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{m}_1, \mathbb{A}) \\ \mathsf{s}_0 \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_i, \mathsf{ct}_0) \\ \mathsf{s}_0 \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_i, \mathsf{ct}_1) \\ \text{output } (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{s}_0, \mathsf{s}_1) \end{pmatrix}$$
$$\overset{\text{Transform}}{\Longleftrightarrow} \begin{pmatrix} \mathsf{k} \leftarrow_{\$} \{0,1\}^{\lambda} \\ \mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{k}, \mathbb{A}) \\ \mathsf{s} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_i, \mathsf{ct}) \\ \mathsf{c}_0 \leftarrow \mathsf{AEAD.Enc}(\mathsf{k}, 0 \parallel \mathsf{m}_0) \\ \mathsf{c}_1 \leftarrow \mathsf{AEAD.Enc}(\mathsf{k}, 1 \parallel \mathsf{m}_1) \\ \text{output } (\mathsf{ct}, \mathsf{s}, \mathsf{c}_0, \mathsf{c}_1) \end{pmatrix}.$$

Necessity of Signatures: Many existing cloud storage systems assume the cloud provider to be semi-honest, in which the ciphertext integrity is not a security concern. However, if we assume the cloud provider to be covert, we need to protect the ciphertext integrity, like using the signatures from the data owner. The signature should also sign on the file metadata, including the file name and the version.

Performance: The security and correctness of the two constructions are the same. But the latter has a benefit in performance – the CP-ABE encryption and decryption, which rely on heavy pairing, are reduced to one. The two piece of the AEAD-encrypted messages can still be sent in arbitrary order.

E. Bloom Filter

Bloom filter [41] is an m-bit sequence for membership test that is reasonably accurate and space-efficient. The bloom filter BF of m-bit for strings in $\{0,1\}^{\mathsf{poly}(\lambda)}$ as follows.

- $\mathsf{bf} \leftarrow \mathsf{Setup}(m, \lambda)$ generates an empty m-bit bit array.
- $\mathsf{bf}' \leftarrow \mathsf{Insert}(\mathsf{bf}, e)$ inserts an element e by setting the following l positions of bf to 1: $H(\mathsf{k}, 1||e)$, $H(\mathsf{k}, 2||e)$, $\cdots$, $H(\mathsf{k}, l||e)$, where $H(\mathsf{k}, \cdot)$ is a keyed collision-resistant hash function and k is a security parameter.
- $b \leftarrow \mathsf{Test}(\mathsf{bf}, e)$ checks whether the element e has been inserted to the bloom filter by checking whether all of these positions $H(\mathsf{k}, 1||e), H(\mathsf{k}, 2||e), \ldots, H(\mathsf{k}, l||e)$ are 1.

Bloom filter has false positives but no false negatives. When bloom filter says that an element is in the set, it may be false. According to the literatures [41]–[43], the false positive rate of a m-bit Bloom filter is:

$$fp \approx \left(1 - \left(1 - \frac{1}{m}\right)^{ln}\right)^{l},$$

where n is the number of the existing members in a set, and l is the number of hash functions used in the bloom filter.

F. Building Systems Against Covert Adversaries

Many of previous encrypted cloud storage schemes focus on semi-honest cloud provider. This assumption is strong, as the cloud storage provider can perform an active attack (e.g., tamper the ciphertext) and may never be caught. However, a maliciously secure scheme against cloud

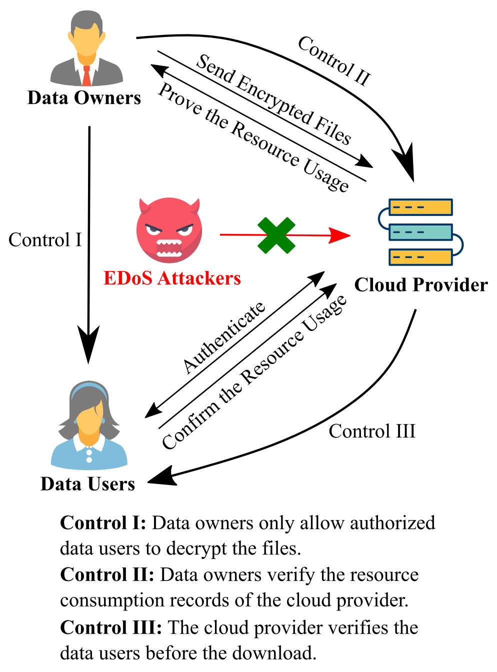

Data Owners
Control II
Send Encrypted Files
Prove the Resource Usage
Control I
EDoS Attackers
Cloud Provider
Authenticate
Confirm the Resource Usage
Control III
Data Users
Control I: Data owners only allow authorized data users to decrypt the files.
Control II: Data owners verify the resource consumption records of the cloud provider.
Control III: The cloud provider verifies the data users before the download.

Fig. 1. System model of the encrypted cloud storage with mitigation of EDoS attacks and transparency of resource consumption accounting.

is too heavy, which brings computation and communication overhead.

Aumann and Lindell [24] describes a trade-off between malicious security and semi-honest security, called *covert security*. This notion is very real-world and practical: big cloud providers (e.g., Microsoft, Google, Amazon) are promising secure execution to customers and shareholders. If any attempt that cloud deviates from the agreed protocol will be caught with a small but reasonable possibility (e.g., $p = 0.001$), the cloud dares not to do so – being caught not only loses the contracts between this customer, but could arouse media concerns that destroy the reputation of the brand. With this insufferable outcome and sufficient deterrence, the cloud server refrains from attacking like this, as we assume.

This requires that *inside the protocol*, there are some verification steps to *catch* a misbehaving cloud. This verification step does not necessarily be as sound as *an argument* defined in proof system, but just probabilistic (e.g., 50% true positive) is sufficient. As the aftermath of being caught for even one time is serious, the verification provides enough deterrence to let the cloud obey the protocol. We formalize the definition of cover security in Section IV-B. The verification steps in our protocol includes Section III-D, VI-C, III-E.

IV. System Model and Security Model

In this section, we first describe the three-party model for cloud storage used in our construction in Section IV-A. Furthermore, the security against malicious data users and a *covert* cloud provider is defined in Section IV-B.

A. System Model

As shown in Fig. 1, the cloud storage system consists of three entities: ***data owners***, ***data users***, and the ***cloud provider***.

- ***Data owners*** are the owner and publisher of files and pay for the resource consumption on file sharing. As the payers for cloud services, the data owners want the transparency of resource consumption to ensure fair billing. The data owners require the cloud provider to justify the resource usage. In our system, the data owner is *not always online*.
- ***Data users*** want to obtain some files from the cloud provider stored on the cloud storage. They need to be authenticated by the cloud provider before the download (to thwart EDoS attacks). The authorized users then confirm (and sign for) the resource consumption for this download to the cloud provider.
- ***Cloud provider*** hosts the encrypted storage and is *always online*. It records the resource consumption and charges data owners based on that record. The cloud is not public-accessible in our system as it has an authentication based access control. Only data users satisfying the access policy can download the corresponding files. The cloud provider also collects the proof of the resource consumption to justify the billing.

As shown in Fig. 1, we have three controls among three entities in our system:

- **Control I.** Data owners assign an access policy in the document, which controls the set of data users who have the privileges to decrypt the contents.
- **Control II.** Data owners verifies the resource consumption from the cloud provider, which controls the cloud provider not to exaggerate the resource usage.
- **Control III.** The cloud provider verifies whether the user can decrypt before the download, which controls the ability of a malicious user who launches DDoS/EDoS attacks.

Moreover, our system differs from previous cloud storage constructions, as we take into account the resource consumption. In practice, the cloud services are usually charged according to the resource consumption, which includes the resource spent on attackers. The DDoS/EDoS attacks will invariably succeed and raise the overhead, which is controlled in our system due to the introduction of the cloud-side access control.

B. Security Assumptions and Requirements

Data owners are trusted and data users can be considered as malicious adversaries. Users may try to cheat for files and launch the EDoS attacks. But authorized users are assumed not to collude with unauthorized users which is impossible to thwart and beyond the scope of this paper.

Different from previous constructions that assume the cloud provider to be a passive attacker (i.e., semi-honest or honest-but-curious) [12], [23], we consider a stronger notation called *covert adversary* [24]. In the real-world cloud service platforms, the cloud providers are usually some big IT enterprises that usually treasure the reputation and try to avoid lawsuits. If any attempt to violate the protocol can be detected by any honest parties (e.g. data owners or data users) with a significant possibility like $p = 0.001$, they dare not to cheat. However, honest parties must keep verifying to keep the deterrence.

Definition 3 (Security Against Covert Adversaries) [24]: Consider a protocol π, the covert security with a parameter ϵ for this protocol π means the following for every covert adversary $\mathcal{A}$:

- $\mathcal{A}$ is caught with a probability of ϵ if $\mathcal{A}$ cheats;
- $\mathcal{A}$ learns nothing in addition compared with honest executions when $\mathcal{A}$ is caught.

To achieve the covert security, the protocol needs some verification steps to *catch* a misbehaving cloud. This verification steps can be probabilistic (e.g., 50% true positive) as this still satisfies the covert security.

We define the two security requirements of our system: 1) Unauthorized data users that do not satisfy the access policy of the file cannot *download* any files; 2) The cloud provider cannot forge a significant proportion of the proofs for resource consumption without being discovered with a sufficient possibility; 3) The protocol can be compatible with some CP-ABE variants with additional privacy guarantee such that the policy and attribute set indistinguishability is not broken.

Definition 4 (Security Against EDoS Attacks): For every security parameter $\lambda \in \mathbb{N}$, every valid access policy $\mathbb{A}$, and every (malicious) PPT user $\widetilde{\mathsf{user}}_i$ whose attribute set $\mathcal{A}_i$ does not satisfy $\mathbb{A}$, there is a constant c independent of filesize, for a single file access session:

$$\Pr\left[\mathsf{server} \text{ sends more than } c \text{ bytes}\right] = \mathsf{negl}(\lambda).$$

Definition 5 (Accounting of Resource Consumption): Consider a forgery proportion $\alpha > 0$ and a threshold detection possibility ϵ, For all security parameter $k \in \mathbb{N}$ where $2^{-\mathsf{k}} < \epsilon$, for a PPT cloud provider in the covert security model, $\widetilde{\mathsf{server}}$ that forges the proofs of resource consumption at least the proportion α,

$$\Pr\left[\mathsf{data\ owner} \text{ knows } \widetilde{\mathsf{server}} \text{ cheats}\right] > \epsilon.$$

Definition 6 (Policy and Attribute Set Indistinguishability): To make the protocol compatible to some CP-ABE variants with additional privacy guarantee, we provide *policy and attribute set indistinguishability* such that no side information is leaked from the protocol to the cloud. Assume the following setup:

$$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABE.Gen}(1^\lambda),\ \mathsf{k} \leftarrow \{0,1\}^\lambda,$$
$$\mathsf{sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, \mathcal{A}_i) \Longrightarrow \mathsf{user},$$
$$\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{k}, \mathbb{A}) \Longrightarrow \widetilde{\mathsf{server}},$$
$$\mathsf{s} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_{owner}}, \mathsf{ct}) \Longrightarrow \widetilde{\mathsf{server}}.$$

We denote the above setup as the hybrid $\mathbf{H}_0$. If $\mathcal{A}_i \in \mathbb{A}$, the cloud server cannot distinguish between that with these two hybrids:

- $\mathbf{H}_1^a$: replaced the attribute set $\mathcal{A}_i$ to an arbitrary attribute set $\mathcal{A}'_i$ from $\mathbf{H}_0$ that satisfies the access policy $\mathbb{A}$.
- $\mathbf{H}_2^a$: replaced the access policy $\mathbb{A}$ to arbitrary access policy from $\mathbf{H}_1^a$ such the size of ct does not change and $\mathcal{A}'_i \in \mathbb{A}$.

If $\mathcal{A}_i \notin \mathbb{A}$, the server cannot distinguish these:

- $\mathbf{H}_1^b$: replaced the attribute set $\mathcal{A}_i$ to an arbitrary attribute set $\mathcal{A}'_i$ from $\mathbf{H}_0$ that does not satisfy the access policy $\mathbb{A}$.
- $\mathbf{H}_2^b$: replaced the access policy $\mathbb{A}$ to arbitrary access policy from $\mathbf{H}_1^b$ such the size of ct does not change and $\mathcal{A}'_i \notin \mathbb{A}$.

V. Our Proposed Scheme

A. Overview of Our Scheme

To achieve the security requirements, the scheme consists of two componetns: 1) A cloud-side access control to block users whose attribute set $\mathcal{A}_i$ does not satisfy the access policy $\mathbb{A}$; 2) A proof-collecting subsystem where the cloud provider can collect the proofs of resource consumption from users, and present to the data owners later.

In real-world scenarios, it is reasonable to specify an expected maximal download times, and data owners can remain offline unless it wants to increase this value. This leads to our first protocol: Partially Outsourced Protocol (POP) (V-B). In some other cases where the data owner cannot set an expectations of download times or would be offline for a long time, the data owner can delegate to the cloud. This leads to our second protocol: Fully Outsourced Protocol (FOP) (V-C).

B. Partially Outsourced Protocol (POP)

In this protocol, the data owner encrypts an ephemeral key in CP-ABE, which is then used for message encryption/decryption and cloud-side access control. The data owner provides the cloud provider with N challenge ciphertexts $\{\mathsf{enchal}_i\}_{i\in[N]}$ and the hashed challenges $\{\mathsf{hash}_i\}_{i\in[N]}$. The user proves the the legitimacy to the cloud provider by showing the decryption result chal_j of the randomly selected unused challenge ciphertext enchal_j is a preimage of hash_j. If the user response is valid, the cloud provider stores the user response for further resource consumption accounting. Furthermore, To boost the efficiency and together reduce the storage space, we introduce the bloom filter for data owners to store their challenge plaintexts. This bloom filter can be stored locally or remotely on the cloud server. As the process of challenge update should be implemented on demand or periodically by the data owner, which cannot be outsouced to the cloud, we call the scheme as Partially Outsouceed Protocol (POP).

The procedure of POP is described in detail as follows:

1) ***Encrypt and Upload (POP-EU)***: This operation is implemented by an individual data owner independently, which can be divided into the following four steps:
   - ***POP-EU-1***: The data owner uses hybrid encryption to encrypt the message. The data owner randomly selects a symmetric key $\mathsf{k} \leftarrow_\$ \{0,1\}^\lambda$ and uses the key to encrypt the message M. Then the data owner encrypts that symmetric key k with CP-ABE under $\mathbb{A}$:

$$\mathsf{c}_0 \leftarrow \mathsf{AEAD.Enc}(\mathsf{k}, \text{“}\textit{message}\text{”} \parallel M),$$
$$\mathsf{c}_1 \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{k}, \mathbb{A}),$$
$$\mathsf{c}_2 \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_{owner}}, \mathsf{c}_1).$$

- ***POP-EU-2***: The data owner randomly generates N challenge plaintexts from the message space. They should be different with each other.

$$\{\mathsf{chal}_1, \mathsf{chal}_2, \ldots, \mathsf{chal}_N\},\ \mathsf{chal} \leftarrow_{\$} \{0,1\}^L.$$

The data owner generates the hashes of these challenges:

$$\mathsf{hash}_i = H(\mathsf{chal}_i), \quad \forall i \in [1, N],$$

where $H(\cdot)$ is a collision-resistant hash function. For each challenge plaintext chal_i, the data owner uses k to encrypt it with a fixed prefix *"challenge"*. The prefix makes these challenges different from messages, which prevents the cloud from deceiving the users into decrypting messages instead of challenges. Here the encryption is also under the same hybrid encryption structure:

$$\mathsf{enchal}_i = \mathsf{AEAD.Enc}(\mathsf{k}, \textit{"challenge"} || \mathsf{chal}_i).$$

Now, we have

$$\mathsf{c}_3 = \{\mathsf{hash}_i\}_{i \in [N]}, \quad \mathsf{c}_4 = \{\mathsf{chal}_i\}_{i \in [N]}.$$

- ***POP-EU-3***: The data owner creates a bloom filter to store the challenge plaintexts. We denote m as the size of the bloom filter.

$$\mathsf{bf} \leftarrow \mathsf{BF.Setup}(m, \lambda), \quad \forall i \in [N],\ \mathsf{bf} \leftarrow \mathsf{BF.Insert}(\mathsf{bf}, \mathsf{chal}_i).$$

And then the data owner encrypts the bloom filter:

$$\mathsf{c}_5 = \mathsf{ABAE.Enc}(\mathsf{k}, \mathsf{bf}),$$

where k is the data owner's secret key. Note that to avoid the cloud provider understanding the structure of the bloom filter, the data owner should use its own **keyed** hash functions in the element insertion and test. We assume that the data owner keeps the version number of the bloom filter to thwart rollback attacks.

- ***POP-EU-4***: The following tuple is uploaded to the cloud:

$$\mathsf{ct} = (\mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4, \mathsf{c}_5).$$

2) ***Cloud-side Access Control: POP-CR***.

- ***POP-CR-1***: The cloud provider selects one of the unused challenge enchal_j and sends the following tuple to the user:

$$(\mathsf{c}_1, \mathsf{c}_2, \mathsf{enchal}_j).$$

The data user decrypts the ciphertexts and verifies the signature of the owner. The decryption of c_1 requires the data user to satisfy the policy $\mathbb{A}$:

$$\textsc{halt} \text{ if } \mathsf{SIG.Verify}(\mathsf{vk}_{\mathsf{owner}}, \mathsf{c}_2, \mathsf{c}_1) = 0,$$
$$\mathsf{k} \leftarrow \mathsf{ABE.Dec}(\mathsf{sk}_i, c_1),$$
$$\mathsf{chal}'_j \leftarrow \mathsf{AEAD.Dec}(\mathsf{k}, \mathsf{enchal}_j).$$

The data user sends chal'_j to the cloud provider.

- ***POP-CR-2***: The cloud checks $\mathsf{hash}_j \overset{?}{=} H(\mathsf{chal}'_j)$. If true, the cloud sends c_0 to the data user, which can be decrypted with the session key k and meanwhile the challenge as *used*. Otherwise, the cloud aborts. The user response chal'_j is the proof of the resource consumption accounting.

3) ***Challenge update (POP-SU)***: If the specified upper bound of download times (N) has not yet reached, there is no need to update. But if the data owner wants to provide additional challenges, either on-demand or periodically, both only needs to be online for a short period, it is also supported. The update process is the same as that in the phase of POP-EU-2 under the same key k. We assume the data owner keeps a record of session keys either in local storage or outsourced to cloud in an encrypted form. As the plaintext space for challenges is sufficiently large, we assume no duplicated challenge plaintexts are generated. The bloom filter (and its encryption form) introduced in POP-EU-3 will be reconstructed.

4) ***Resource Accounting (POP-RA)***: data owners and the cloud interactively implement this operation. The cloud sends back the encrypted bloom filter c_5 and m user responses $\{\mathsf{chal}_i\}_{i=1,2,\ldots,m}$. Given the probabilistic check rate β, $\beta \cdot m$ responses are randomly selected for verification:

$$(\mathsf{chal}'_1, \mathsf{chal}'_2, \ldots, \mathsf{chal}'_{\beta m}) \leftarrow_{\$} (\mathsf{chal}_1, \mathsf{chal}_2, \ldots, \mathsf{chal}_m).$$

The data owner decrypts the bloom filter c_5, only if integrity holds and the version number indicates the freshness, the data owner can accept the resource consumption if:

$$\sum_{i=1}^{\beta \cdot m} \mathsf{BF.Test}(\mathsf{bf}, \mathsf{chal}'_i) = \beta \cdot m.$$

Though the bloom filter has some false positives, it is sufficient to achieve the covert security against a cloud provider, as we defined in Section IV-B.

C. Fully Outsourced Protocol (FOP)

If we cannot expect the file download times, we can outsource the challenge update to the cloud. In this section, we give a protocol based on signature algorithm, which has both the outsourced challenges generation/update and resource accounting without an external PKI, therefore we name it as Fully Outsourced Protocol (FOP).

Compared with POP, we have two main differences: 1) Instead of having the data owners generate the challenges $\{\mathsf{enchal}_i\}_{i \in [N]}$, the challenges are generated by the cloud; 2) The data owners generate a pair of signature keys $(\mathsf{vk}, \mathsf{sk})$ for every file, with which legitimate users sign a confirmation to prove the resource consumption.

The main procedure of FOP is described as follows:

1) ***Encrypt and Upload (FOP-EU):***

- ***FOP-EU-1***: This operation is the same as POP-EU-1.

- ***FOP-EU-2***: The data owner generates a signing key pair:

$$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.Gen}(1^\lambda),$$

We avoid the external PKI by only using the basic primitive of digital signatures directly. The signing key sk is encrypted under k:

$$\mathsf{c}_3 = \mathsf{AEAD.Enc}\,(\mathsf{k}, \text{“}signing\text{”} \parallel \mathsf{sk}).$$

We provide the verifying key and the collision-resistant hash of k to the cloud:

$$\mathsf{c}_4 = H(\mathsf{k}),\ \ \mathsf{c}_5 = \mathsf{vk}.$$

- ***FOP-EU-3***: The following tuple is uploaded to the cloud:

$$\mathsf{ct} = (\mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4, \mathsf{c}_5).$$

2) ***Outsourced Challenge Generation (FOP-CG)***: In FOP, the cloud provider generates the challenges, which is different from POP. The generation can be done in advance or on demand. We choose the former.
The challenge is encrypted by c_4 instead of k:

$$\mathsf{chal}_i \xleftarrow{\$} \{0,1\}^\lambda,\ \ \mathsf{enchal}_i = \mathsf{AEAD.Enc}\,(\mathsf{c}_4, \mathsf{chal}_i).$$

3) ***Challenge-Response (FOP-CR)***. Data owners and the cloud run this operation, which can divided into the following 2 steps:

- ***FOP-CR-1***: The cloud provider selects an unused challenge enchal_j and sends the following to the user:

$$\left(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{enchal}_j\right).$$

As the data user has the CP-ABE secret key sk_i, he/she can verify the signature and decrypts these ciphertexts:

$$\begin{aligned}&\textsc{halt}\text{ if } \mathsf{SIG.Verify}\,(\mathsf{vk}_{\mathsf{owner}}, \mathsf{c}_2, \mathsf{c}_1) = 0,\\ &\mathsf{k} \leftarrow \mathsf{ABE.Dec}\,(\mathsf{sk}_i, \mathsf{c}_1)\,,\\ &\mathsf{c}_4 \leftarrow H(\mathsf{k}),\\ &\mathsf{chal}'_j \leftarrow \mathsf{AEAD.Dec}\left(\mathsf{k}, \mathsf{enchal}_j\right),\\ &\mathsf{sk} \leftarrow \mathsf{AEAD.Dec}\,(\mathsf{k}, \mathsf{c}_3).\end{aligned}$$

And the data user generates the proof by making the signature with the signing key sk, which is generated by the data owner:

$$\mathsf{prof} \leftarrow \left(\mathsf{SIG.Sign}\left(\mathsf{sk}, \mathsf{chal}_j \| \mathsf{Info}\right), \mathsf{chal}_j, \mathsf{Info}\right),$$

where Info is an auxiliary information that consists of the timestamp and the filename.

- ***FOP-CR-2***: The cloud checks whether the user responses is the correct chal_j and whether the signed record prof is valid. If true, the cloud sends c_0 to the user, otherwise it aborts.

4) ***Resource Accounting (FOP-RA)***. This operation is interactively implemented by the data owner and the cloud. The data owner asks the cloud to send all signed records $\{\mathsf{prof}_i\}_{i=1,2,\ldots,m}$. Given the probabilistic check rate β, $\beta \cdot m$ responses are randomly chosen for verification.

$$(\mathsf{prof}'_1, \mathsf{prof}'_2, \ldots, \mathsf{prof}'_{\beta m}) \leftarrow_{\$} (\mathsf{prof}_1, \mathsf{prof}_2, \ldots, \mathsf{prof}_m).$$

The data owner accepts the resource consumption if:

$$\sum_{i=1}^{\beta \cdot m} \mathsf{SIG.Verify}\,(\mathsf{vk}, \mathsf{chal}_i[0], \mathsf{chal}_i[1] \parallel \mathsf{chal}_i[2]) = \beta \cdot m.$$

The probabilistic inspection remains sufficient deterrence as we will analysis later in Section VI-C.

VI. Security Analysis

In this section, we analyze the two protocols on how they achieves several main security properties. The security requirements are listed in Section IV-B. We describe the security against EDoS attacks (in Section VI-A), and the resource consumption accounting (in Section VI-B, VI-C, VI-D). Then we show the covert security against the cloud provider, including the active attacks (in Section VI-E) and passive attacks (in Section VI-F).

A. Security Against EDoS Attacks

EDoS attackers are those that do not satisfy the access policy (i.e., unauthorized users) but want to trigger the cloud provider to send something through the network, as a result the resource consumption increases. To thwart such attacks, the cloud provider uses authentication. The protocols only send a constant amount of bytes to the data user before it passes the cloud-side access control. To succeed a EDoS attack in our definition, the attacker has to first pass the cloud-side access control. We want to show that for any PPT user, there is a constant c independent of the file size, such that:

$$\Pr\left[\mathsf{server}\text{ sends more than } c \text{ bytes}\right] = \mathsf{negl}(\lambda).$$

Security of POP: We use Random Oracle (RO) heuristic and hybrid arguments. The input to the PPT user is:

$$\begin{aligned}\mathsf{c}_1 &= \mathsf{ABE.Enc}\,(\mathsf{mpk}, \mathsf{k}, \mathbb{A})\,,\\ \mathsf{c}_2 &= \mathsf{SIG.Sign}\,(\mathsf{sk}_{\mathsf{owner}}, \mathsf{c}_1)\,,\\ \mathsf{enchal}_j &= \mathsf{AEAD.Enc}\left(\mathsf{k}, \mathsf{chal}_j\right).\end{aligned}$$

The user needs to output y such that $H(y) = H(\mathsf{chal}_j)$. Consider the input above to be the hybrid $\mathbf{H}_0$ and consider the following hybrids:

$\triangleright$ **Hybrid** $\mathbf{H}_1$. It replaces k in c_1 to 1^λ from $\mathbf{H}_0$. Due to the message indistinguishability of CP-ABE (IND-CPA), we have $\mathbf{H}_0 \overset{\mathsf{c}}{\approx} \mathbf{H}_1$.

$\triangleright$ **Hybrid** $\mathbf{H}_2$. It replaces chal_j to 1^λ in enchal_j from $\mathbf{H}_1$. Due to the message indistinguishability of AEAD (IND-CCA2), $\mathbf{H}_1 \overset{\mathsf{c}}{\approx} \mathbf{H}_2$.

The possibility of guessing a correct y to satisfy the above condition is negligible. In the hybrid $\mathbf{H}_2$, the input provides nothing about chal_j. In the random oracle model, it is negligible that a PPT adversary can make a collision. From the random oracle heuristic, no PPT user wins in $\mathbf{H}_0$ with a sufficient possibility.

Security of FOP: Similarly, the input to the user is:

$$\begin{aligned} \mathsf{c}_1 &= \mathsf{ABE.Enc}\left(\mathsf{mpk}, \mathsf{k}, \mathbb{A}\right), \\ \mathsf{c}_2 &= \mathsf{SIG.Sign}\left(\mathsf{sk}_{\mathsf{owner}}, \mathsf{c}_1\right), \\ \mathsf{c}_3 &= \mathsf{AEAD.Enc}\left(\mathsf{k}, \text{“}\textit{signing}\text{”} \parallel \mathsf{sk}\right), \\ \mathsf{enchal}_j &= \mathsf{AEAD.Enc}\left(H(\mathsf{k}), \mathsf{chal}_j\right). \end{aligned}$$

The attacker needs to output y such that $y = \mathsf{chal}_j$. Consider the input above to be the hybrid $\mathbf{H}_0$. Now consider the following hybrids:

▷ **Hybrid $\mathbf{H}_1$**. It replaces k in c_1 to 1^λ from $\mathbf{H}_0$. We have $\mathbf{H}_0 \overset{\mathsf{c}}{\approx} \mathbf{H}_1$.

▷ **Hybrid $\mathbf{H}_2$**. It replaces (“*signing*” $\parallel$ sk) in c_3 to a random element in $\{0,1\}^{|\mathsf{sk}|+7}$ from $\mathbf{H}_1$. Similarly, we have $\mathbf{H}_1 \overset{\mathsf{c}}{\approx} \mathbf{H}_2$.

▷ **Hybrid $\mathbf{H}_3$**. In the random oracle heuristic, $H(\mathsf{k})$ can be considered as a random function. We can replace chal_j by 1^λ from $\mathbf{H}_2$. We have $\mathbf{H}_2 \overset{\mathsf{c}}{\approx} \mathbf{H}_3$.

The possibility of guessing a correct y to satisfy the above condition is negligible. Because no information about chal_j is left in $\mathbf{H}_3$. From the random oracle heuristic, no PPT user can make EDoS attack to $\mathbf{H}_0$ with a sufficient possibility.

B. Resource Consumption Accounting

As shown in Section VI-A, for a user whose attribute set $\mathcal{A}_i$ does not satisfy the access policy $\mathbb{A}$: 1) The user cannot output a valid preimage chal_j in POP; 2) The user cannot obtain the signing key sk in FOP.

Without the loss of generality, we assume that the cloud provider has never been granted any attributes and doesn't collude with any authorized users. The result above can also be applied to the cloud provider as follows:

- In POP, to forge a proof, the cloud provider needs to decrypt enchal_i without the key k or find a preimage of hash_i.
- In FOP, to forge a proof, the cloud provider needs to generate a valid signed record prof on a message that has never been signed, without knowing the signing key sk.

We now show that no PPT algorithm $\mathcal{A}$ can forge:

Unforgeability in POP: The input to the algorithm $\mathcal{A}$ is the following (denoted as the hybrid $\mathbf{H}_0$):

$$\begin{pmatrix} \mathsf{enchal}_i = \mathsf{AEAD.Enc}\left(\mathsf{k}, \text{“}\textit{challenge}\text{”} \parallel \mathsf{chal}_i\right), \\ \mathsf{hash}_i = H(\mathsf{chal}_i). \end{pmatrix}$$

▷ **Hybrid $\mathbf{H}_1$.** It replaces chal_i in enchal_i with 1^λ from $\mathbf{H}_0$. Due to IND-CCA2 of AEAD, we have $\mathbf{H}_0 \overset{\mathsf{c}}{\approx} \mathbf{H}_1$.

▷ **Hybrid $\mathbf{H}_2$.** Replaced $H(\cdot)$ with the random oracle $\mathsf{RO}(\cdot)$. Due to the random oracle heuristic, we have $\mathbf{H}_1 \overset{\mathsf{c}}{\approx} \mathbf{H}_2$. The possibility of guessing the preimage of a random oracle is negligible. This means the proof is unforgeable.

Unforgeability in FOP: The input to the algorithm $\mathcal{A}$ is the verifying key vk. Due to the existential unforgeability of the signature scheme SIG (in Section III-C), no PPT algorithm can forge a signature on messages that have never been signed.

Since it is difficult to forge a proof either in POP or FOP, we only need to consider the bllom filter and the probabilistic check, which can reduce the overhead, but introduce a small possibility for the cloud provider to cheat without being caught.

Bloom Filter (for POP): Note that the data owner uses a keyed collision-resistant hash function as described in Section III-E, where the key is only known to the data owner. No adversary knows the mapping in the bloom filter. And the cloud provider does not know the elements in the bloom filter because it is encrypted. We want to show that even if the cloud provider knows N' elements in the bloom filter, the cloud provider cannot forge an element that passes the test better than a random guess. Consider the input to the PPT adversary $\mathcal{A}$:

$$\{\mathsf{chal}_i\}_{i\in[N']} \text{ where } \mathsf{BF.Test}\left(\mathsf{bf}, \mathsf{chal}_i\right) = 1.$$

The adversary should output chal' which holds the following condition with a significantly higher possibility than a random guess:

$$\mathsf{BF.Test}\left(\mathsf{bf}, \mathsf{chal}'\right) = 1.$$

We denote the above input as hybrid $\mathbf{H}_0$. Consider these hybrids:

▷ **Hybrid $\mathbf{H}_1$.** It replaces the bloom filter scheme BF to BF' where the keyed CRH function $H(\mathsf{k}\cdot)$ is replaced by queries to the random oracle $\mathsf{RO}(\mathsf{k} \parallel \cdot)$. Because of the random oracle heuristic, we have $\mathbf{H}_0 \overset{\mathsf{c}}{\approx} \mathbf{H}_1$.

▷ **Hybrid $\mathbf{H}_2$.** The same as $\mathbf{H}_1$. The possibility that the proof hits a positive is:

$$\Pr\left[\text{forging an element to match } \mathsf{BF}\right] \approx \left(1 - \left(1 - \frac{1}{m}\right)^{ln}\right)^{l} \triangleq \delta_{\mathsf{BF}}.$$

Selective Check (for Both POP and FOP): If only a proportion β of all proofs are chosen randomly for resource consumption accounting, it is possible that those proofs forged by the cloud provider are not detected. If the cloud provider forges only one proof, in which the cloud only has limited benefit, the possibility of cheating without being detected is $(1-\beta)$.

C. Sufficient Detection Rate from Probabilistic Check

The probabilistic check reduces the overhead of verification and maintains the covert security. Consider the cloud provider forges a proportion α of proofs and the data owner checks with the proportion β. The detection possibility is as follows:

$$p = \begin{cases} 1 & \text{if } \beta \geq 1-\alpha, \\ 1 - \binom{n(1-\alpha)}{n\beta} / \binom{n}{n\beta} & \text{if } \beta < 1-\alpha. \end{cases}$$

If the cloud forges one invalid proof, the possibility of being detected is at least:

$$p \geq 1 - \binom{n}{(n+1)\beta} / \binom{n+1}{(n+1)\beta} = \beta = O(1),$$

which is sufficient for convert security if hen the data owner selects a non-negative constant β (like 0.001).

To demonstrate how probabilistic check reduces the bandwidth, we give an example: Suppose there are $N = 1000$

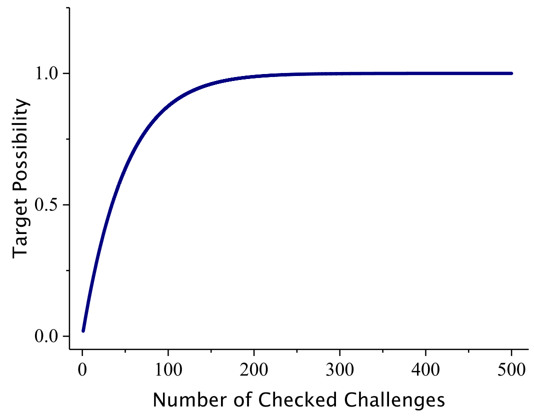


Fig. 2. Probabilistic check example: Number of checked challenges and the target possibility p^{target} of catching the misbehaviors when the cloud is faking 20 proofs ($\alpha = 0.02$) over 1000 authenticated proofs.

challenges and the cloud wants to forge $N' = 20$ ($\alpha = 0.02$) proofs. We select different $\beta \cdot (N + N')$ and calculate a possibility p^{target} catching the misbehaviors:

$$1 - \binom{N}{(N+N')\cdot\beta} \Big/ \binom{N+N'}{(N+N')\cdot\beta} \geq p^{\text{target}}.$$

This result is shown in Fig. 2. From the definition of covert security, having a detection probability of 10% is sufficient to force the cloud provider t behave, while setting $\beta = 0.1\%$ (checking 1 out of 1020 proofs) has 2% catching possibility. Small increase of β can lead to high catching possibility as $\beta = 0.6\%$ (checking 6 out of 1020 proofs) has 10% catching possibility. If the cloud wants to generate many fake proofs to charge significantly more money, the catching possibility increases.

D. Sufficient Detection Rate from Bloom Filter

For reducing the storage of challenge plaintexts in POP, we use the bloom filter. We want to show the false positive rate is sufficient so that secure against covert security. Assume the number of hash functions is $l = \frac{m}{n} \cdot \ln 2$ [44], and the proportion m/n is a non-negative constant γ, we have:

$$fp \approx (0.5^{\ln 2})^{\gamma} > \frac{1}{2^{\gamma}} > 0.$$

As γ is a non-negative constant, we guarantee that the bloom filter is within the definition of covert security.

To evaluate how bloom filter can reduce the storage, we assume there are N challenges. The length of plaintext is 128 bits. We parameterize the minimal length of bloom filter m to achieve a false positive not larger than 90%.

$$(1 - (1 - \frac{1}{m})^{lN})^{l} \leq 90\%.$$

The result is shown in Fig 3. The lossy compression ratio of this bloom filter construction is $\frac{m}{N \cdot 128} \approx 0.34\%$.

E. Covert Security Against the Cloud Provider

We want to achieve the covert security introduced in Section III-F that the cloud provider will be caught with a high possibility if the cloud want to cheat.

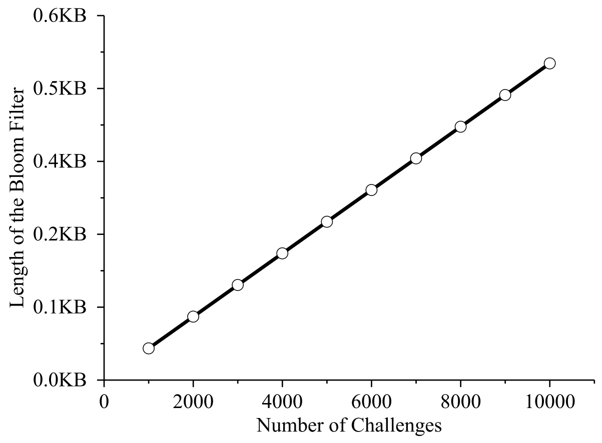


Fig. 3. Bloom filter compression: Number of challenges and the length of bloom filter (in plaintext) for storage with 90% false positive.

For the cloud provider to cheat in the accounting (assume *only one* forged proof), as Section VI-B said, the possibility to being caught is:

$$\text{POP: } \epsilon = \beta \cdot (1 - \delta_{\text{bf}}),$$
$$\text{FOP: } \epsilon = \beta.$$

If we set $\beta_{\text{POP}} = 10\%$ and $fp_{\text{POP}} = 90\%$, the catching possibility of POP is 1% which is reasonable for covert security. If we set $\beta_{\text{FOP}} = 1\%$, the catching possibility of FOP is 1%.

If the cloud forge more proofs, it becomes easier to be caught. In summary, the covert security is attainable with little verification.

F. Policy and Attribute Set Indistinguishability

To make our protocols compatible with some existing CP-ABE schemes and variants [12], [16] that provide additional privacy guarantee, the cloud-side access-control should reveal nothing about the policy and attribute set, except whether the user satisfies the access policy.

In POP, if the user's attribute set satisfies the access policy ($\mathcal{A}_i \in \mathbb{A}$). The user does not halt only if the ciphertext has a valid signature. Regardless of the access policy and attribute sets, the user computes the results using the same key k, which is the only plaintext of the signed ciphertext. We have $\mathbf{H}_0 \overset{c}{\approx} \mathbf{H}_1^a \overset{c}{\approx} \mathbf{H}_2^a$.

If the user does not satisfy the access policy, the user halts after decrypting the CP-ABE ciphertext, which is signed by the data owner, and doesn't respond. We have $\mathbf{H}_0 \overset{c}{\approx} \mathbf{H}_1^b \overset{c}{\approx} \mathbf{H}_2^b$.

In FOP, the same reasons show the indistinguishability in both cases ($\mathcal{A}_i \in \mathbb{A}$ or $\mathcal{A}_i \notin \mathbb{A}$). This shows that our scheme yields policy and attribute set indistinguishability.

VII. Performance Analysis

In this section, We give the experiment setup and analyze the computation overhead and communication overhead between original CP-ABE based storage (without covert security), POP, and FOP, respectively.

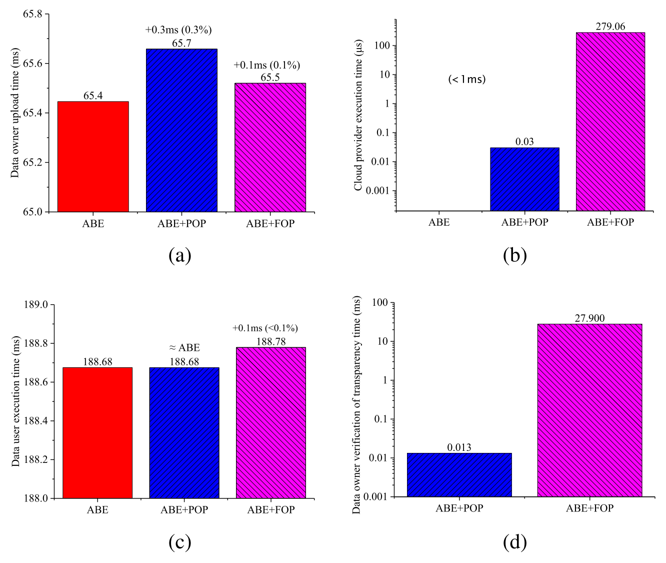


Fig. 4. Performance analysis of the computation cost with the illustration of the communication under attacks. (a) Data owner upload time. (b) Cloud provider execution time. (c) Data user execution time. (d) Verification of transparency time.

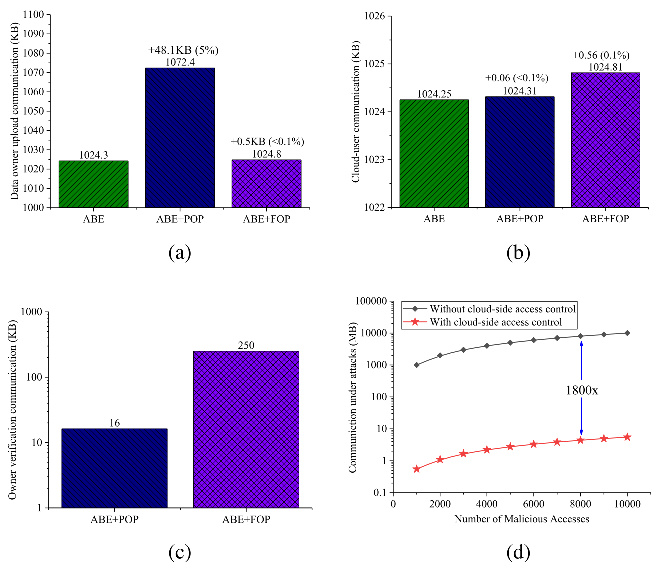


Fig. 5. Performance analysis of the communication cost with the illustration of the communication under attacks. (a) Data owner upload communication. (b) Cloud-user communication. (c) Owner verification communication. (d) Communication under attacks.

A. Experiment Setup

The experiment uses Microsoft Azure DS1_V2 (Single Core, 3.5GB Memory, SSD) running OpenSSL 1.1.0 and cpabe toolkit 0.11 [9].

The encrypted cloud storage has 2^{20} files each of length 1MB. The cpabe library uses a1 elliptic curve, which has 2048-bit discrete-log-equivalent security. The access structure of CP-ABE is a 5-attribute-AND policy for illustration. The AEAD encryption is implemented with AES-GCM-128 and the secure hash function is implemented with AES-NI. The key lengths for ECDSA and RSA are 160 bit and 4096 bit, respectively.

The catching possibility is set to be $\epsilon = 10\%$, both for POP and FOP. In POP, we trade-off the verification time and the length of bloom filter by requiring the compression rate be $\approx 5\%$ (i.e., $k = 4$ and $m = 5021$), and the $\beta_{\mathsf{POP}} = 11\%$ The number of challenges is $N = 1000$ each file in POP. FOP generates the challenge upon user request, and set the probabilistic check rate $\beta_{\mathsf{FOP}} = 10\%$.

B. Computation Overhead

The experimental result in terms of computation overhead is given in Fig. 4(a), 4(b), 4(c), and 4(d). Since the CP-ABE has bilinear pairing, the encryption and the decryption costs 64ms and 188ms in our experiment, respectively. The overhead of our construction over CP-ABE is from: 1) The encryption and hash for generating $N = 1000$ challenges and creating the bloom filter BL in POP; 2) The key generation, signature, and verification from ECDSA in FOP.

For the computation overhead, when the data owner uploads the file (as shown in Fig. 4(a)), POP and FOP has 0.3ms and 0.1ms additional execution time, respectively. The addition is small compared with the original ABE (with owner's signature), as is <0.5% in Fig. 4(a).

For the computation overhead, when the cloud provider authenticates a data user (as shown in Fig. 4(b)), POP and FOP brings an additional overhead, 0.03μs and 279.06μs, respectively. This overhead is less than 1ms and smaller than the LAN network latency. Note that when the user is not eligible for access, the time cost is even less for FOP, as in FOP-CR-2, the cloud checks the challenge plaintext first, and then the signature. If the user cannot provide the challenge plaintext, the cloud does not need to verify the signature, which makes the overhead very small.

When an authorized data user retrieves a file (as shown in Fig. 4(c)), the data user needs to solve the cloud provides's challenge. The challenge decryption can be done within several symmetric encryption and hashing, which is efficient both in POP and FOP. In FOP, the data user needs to generate a proof from ECDSA, which is small (<0.1%) compared with CP-ABE decryption. This shows the impact to honest users is small.

For the resource consumption accounting (as shown in Fig. 4(d)), the time of verification is less than 100ms, for verifying a total of ≤ 1000 challenges. This is only necessary when the data owner who wants to account the resource consumption.

C. Communication Overhead

The experiment result in terms of communication overhead is given in Fig. 5(a), 5(b), and 5(c). Mitigation against EDoS attacks is illustrated in Fig. 5(d) on how the bandwidth is saved when under the EDoS attacks. The proportion of the additional communication overhead in our construction relates to the uploaded file length (e.g., 1MB in the experiment). When the file is larger, the overhead is less significant.

For data owner upload communication (as shown in Fig. 5(a)), the increase for POP is noticeable ($\approx$ 5%)

for 1MB file for the challenge generation. If the number of challenges is large or is hard to anticipate, the data owners can use FOP.

For the communication overhead between the cloud provider and *honest* users for file download and cloud-side access control (as shown in Fig. 5(b)), the addition is $<$1KB that is small when the file length is 1MB. The ECDSA signature scheme also gives a short signature which reduces the cost.

Finally, the purpose of our cloud-side access control is to reduce the communication under EDoS attacks. When there are malicious requests, our cloud-side access control can reduce the communication overhead to 1/1800 if the file size is 1MB (as shown in Fig. 5(d)). It indicates that this access control thwarts the EDoS attacks.

VIII. Conclusion

In this paper, we propose a combined the cloud-side and data owner-side access control in encrypted cloud storage, which is resistant to DDoS/EDoS attacks and provides resource consumption accounting. Our system supports arbitrary CP-ABE constructions. The construction is secure against malicious data users and a covert cloud provider. We relax the security requirement of the cloud provider to covert adversaries, which is a more practical and relaxed notion than that with semi-honest adversaries. To make use of the covert security, we use bloom filter and probabilistic check in the resource consumption accounting to reduce the overhead. Performance analysis shows that the overhead of our construction is small over existing systems.

Acknowledgment

The authors sincerely thank the anonymous referees for their invaluable suggestions that have led to the present improved version of the original manuscript.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[2] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[3] L. Zhou, Y. Zhu, and A. Castiglione, "Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner," *Comput. Secur.*, vol. 69, pp. 84–96, Aug. 2017.

[4] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing SIFT: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3411–3425, Jul. 2016.

[5] H.-M. Sun, Y.-H. Chen, and Y.-H. Lin, "OPass: A user authentication protocol resistant to password stealing and password reuse attacks," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 651–663, Apr. 2012.

[6] L. Harn and J. Ren, "Generalized digital certificate for user authentication and key establishment for secure communications," *IEEE Trans. Wireless Commun.*, vol. 10, no. 7, pp. 2372–2379, Jul. 2011.

[7] V. Sekar and P. Maniatis, "Verifiable resource accounting for cloud computing services," in *Proc. 3rd ACM Workshop Cloud Comput. Secur. Workshop*, 2011, pp. 21–26.

[8] C. Chen, P. Maniatis, A. Perrig, A. Vasudevan, and V. Sekar, "Towards verifiable resource accounting for outsourced computation," *ACM SIGPLAN Notices*, vol. 48, no. 7, pp. 167–178, 2013.

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

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

[11] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[12] S. Yu, K. Ren, and W. Lou, "Attribute-based content distribution with hidden policy," in *Proc. 4th Workshop Secure Netw. Protocols (NPSec)*, Oct. 2008, pp. 39–44.

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

[14] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1484–1496, May 2016.

[15] J. Hong *et al.*, "TAFC: Time and attribute factors combined access control for time-sensitive data in public cloud," *IEEE Trans. Serv. Comput.*, 2017. [Online] Avaliable: https://doi.org/10.1109/TSC.2017.2682090

[16] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 35–45, Jun. 2016.

[17] J. Idziorek and M. Tannian, "Exploiting cloud utility models for profit and ruin," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2011, pp. 33–40.

[18] J. Idziorek, M. F. Tannian, and D. Jacobson, "The insecurity of cloud utility models," *IT Prof.*, vol. 15, no. 2, pp. 22–27, Mar./Apr. 2013.

[19] M. H. Sqalli, F. Al-Haidari, and K. Salah, "EDoS-shield—A two-steps mitigation technique against EDoS attacks in cloud computing," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2011, pp. 49–56.

[20] N. Vlajic and A. Slopek, "Web bugs in the cloud: Feasibility study of a new form of EDoS attack," in *Proc. Globecom Workshops (GC Wkshps)*, Mar. 2014, pp. 64–69.

[21] R. K. L. Ko *et al.*, "TrustCloud: A framework for accountability and trust in cloud computing," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2011, pp. 584–588.

[22] D. Ó. Coileáin and D. O'Mahony, "Accounting and accountability in content distribution architectures: A survey," *ACM Comput. Surv.*, vol. 47, no. 4, 2015, Art. no. 59.

[23] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[24] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *Proc. 4th Theory Cryptogr. Conf.*, 2007, pp. 137–156.

[25] I. Damgård, M. Geisler, and J. B. Nielsen, "From passive to covert security at low cost," in *Proc. 7th Theory Cryptogr. Conf.*, 2010, pp. 128–145.

[26] P. Mohassel and E. Weinreb, "Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries," in *Proc. Annu. Int. Conf. Cryptol.*, 2008, pp. 481–496.

[27] V. Goyal, P. Mohassel, and A. Smith, *Efficient Two Party and Multi Party Computation Against Covert Adversaries*. Berlin, Germany: Springer, 2008, pp. 289–306.

[28] F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan, "Sieve: Cryptographically enforced access control for user data in untrusted clouds," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement.*, 2016, pp. 611–626.

[29] K. Xue, J. Hong, Y. Xue, D. S. L. Wei, N. Yu, and P. Hong, "CABE: A new comparable attribute-based encryption construction with 0-encoding and 1-encoding," *IEEE Trans. Comput.*, vol. 66, no. 9, pp. 1491–1503, Sep. 2017.

[30] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advance Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2005, pp. 457–473.

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

[32] K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in *Proc. INFOCOM*, Apr. 2013, pp. 2895–2903.

[33] K. Xue *et al.*, "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 953–967, Apr. 2017.

[34] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Comput. Surv.*, vol. 39, no. 1, 2007, Art. no. 3.

[35] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat DDoS attacks in clouds?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2245–2254, Sep. 2014.

[36] Q. Chen, W. Lin, W. Dou, and S. Yu, "CBF: A packet filtering method for DDoS attack defense in cloud environment," in *Proc. IEEE 9th Int. Conf. Dependable, Autonomic Secure Comput. (DASC)*, Dec. 2011, pp. 427–434.

[37] C. Hoff. *Cloud Computing Security: From DDoS (Distributed Denial of Service) to EDoS (Economic Denial of Sustainability)*. Accessed: Mar. 6, 2018. [Online]. Available: http://www.rationalsurvivability.com/blog/?p=66

[38] J. Idziorek, M. Tannian, and D. Jacobson, "Attribution of fraudulent resource consumption in the cloud," in *Proc. 5th IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2012, pp. 99–106.

[39] D. A. McGrew and J. Viega, "The security and performance of the Galois/counter mode (GCM) of operation," in *Proc. 5th Int. Conf. Cryptol. India*, 2004, pp. 343–355.

[40] J. Katz and M. Yung, "Unforgeable encryption and chosen ciphertext secure modes of operation," in *Proc. 7th Int. Workshop Fast Softw. Encryption*, 2000, pp. 284–299.

[41] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

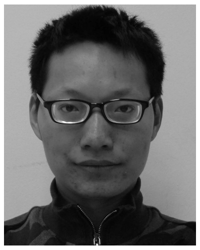
[42] Y. Qiao, T. Li, and S. Chen, "Fast Bloom filters and their generalization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 93–103, Jan. 2014.

[43] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.

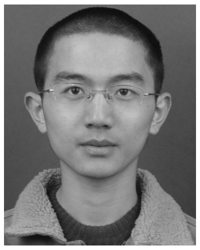
[44] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 131–155, 1st Quart., 2012.

Kaiping Xue (M'09–SM'15) received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and received the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. He is currently an Associate Professor with the Department of Information Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks and network security.

Weikeng Chen received the B.S. degree from the Department of Information Security, University of Science and Technology of China, in 2017. His research interests include applied cryptography and secure multi-party computation.



Wei Li received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2014, and the master's degree from the Department of Electronic Engineering and Information Science, USTC, in 2017. His research interests include network security protocol design and analysis.



Jianan Hong received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2012. He is currently pursuing the Ph.D. degree in information security with the Department of Electronic Engineering and Information Science, USTC. His research interests include secure cloud computing and mobile network security.

Peilin Hong received the B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986. She is currently a Professor and an Advisor for Ph.D. candidates with the Department of EEIS, USTC. She has published two books and over 150 academic papers in several journals and conference proceedings. Her research interests include next-generation Internet, policy control, IP QoS, and information security.