Fully Accountable Data Sharing for Pay-As-You-Go Cloud Scenes

Ti Wang, Hui Ma, Yongbin Zhou, Rui Zhang, and Zishuai Song

Abstract—Many enterprises and individuals prefer to outsource data to public cloud via various pricing approaches. One of the most widely-used approaches is the pay-as-you-go model, where the data owner hires public cloud to share data with data consumers, and only pays for the actually consumed services. To realize controllable and secure data sharing, ciphertext-policy attribute-based encryption (CP-ABE) is a suitable solution, which can provide fine-grained access control and encryption functionalities simultaneously. But there are some serious challenges when applying CP-ABE in pay-as-you-go. Firstly, the decryption cost in ABE is too heavy for data consumers. Secondly, ABE ciphertexts probably suffer distributed denial of services (DDoS) attacks, but there is no solution that can eliminate the security risk. At last, the data owner should audit resource consumption to guarantee the transparency of charge, while the existing method is inefficient. In this work, we propose a general construction named fully accountable ABE (FA-ABE), which simultaneously solves all the challenges by supporting all-sided accountability in the pay-as-you-go model. We formally define the security model and prove the security in the standard model. Also, we implement an instantiate construction with the self-developed library libabe. The experiment results indicate the efficiency and practicality of our construction.

Index Terms—Cloud computing, pay-as-you-go model, data sharing, attribute-based encryption, DDoS attacks.

1 Introduction

Cloud computing is a quite fascinating service paradigm where the computation and storage are moved away from terminal devices to the cloud. Cloud service providers, e.g., Amazon S3, Google Cloud, offer the flexible pricing method, pay-as-you-go, which is one of the most widely-used approaches for enterprises and individuals. The cloud hirer only needs to pay for the consumed services, such as storage, request and data transfer. It is similar to pay for utilities like water and electricity, without requiring long-term contracts or complex licensing.

In order to securely share data on public cloud, data owners need to outsource data in encrypted forms. Ciphertext-policy attribute-based encryption (CP-ABE), which can provide fine-grained access control and encryption functionalities simultaneously [1]–[3], is a suited solution for controllable and secure data sharing in the pay-as-you-go model. Specifically, every participant is assigned with a secret key, which contains several attributes based on his/her role. The data owner encrypts a file associated with an access policy before uploading to the hired cloud, and a data consumer can decrypt only if the attribute set of the secret key satisfies the access policy of the ciphertext. We denote the access control from the data owner to data consumers as Audit 1 in this paper.

Moreover, other two kinds of audits should be carefully considered when applying CP-ABE in the pay-as-you-go cloud scenes [4]: (1) Every data consumer is allowed to request and download any ABE ciphertexts of his/her interest, no matter s/he has the decryption permission. Consequently, frequent requests from data consumers may occupy a large amount of bandwidth, even making the whole cloud system paralyzed. We denote the data download control conducted from the cloud to data consumers as Audit 2. (2) The data owner should be able to verify the resource consumption recorded and claimed by the cloud provider, making sure the charge is transparent and reasonable. The audit from the data owner to the cloud is denoted as Audit 3. Though many schemes have been proposed to achieve fine-grained data sharing for public cloud scenes [4]–[7], there are still three serious challenges when applying CP-ABE in the pay-as-you-go model.

Low Efficiency of ABE. CP-ABE provides powerful and flexible access control, while the communication and computation cost of decryption is quite high. In most of the existing schemes, the ciphertext length, as well as the pairing and exponentiation operations during the decryption phase grow linearly with the complexity of the access policy. It severely limits consumers for usage and essentially impedes ABE from wide-range deployment. Therefore, the first challenge is *how to optimize the communication and computation efficiency of ABE in the pay-as-you-go model?*

DDoS Attacks on ABE Ciphertexts. In most of the CP-ABE schemes, since the cloud provider cannot judge whether the data consumer has the decryption permission of the ciphertext, it will respond to all download requests. Thus unauthorized consumers who cannot actually decrypt the ciphertext, may frequently raise requests to the cloud. Besides, when the computer gets infected by a virus, authorized consumers may request the same ciphertext many times in a short period. We denote these attacks as the "Distributed denial of services (DDoS) attacks on ciphertexts". In the practical application scenario with ABE, files are usually encrypted in the Key/Data Encapsulation Mechanism (KEM/DEM) setting [8], where the ABE ciphertext (as the KEM part)

T. Wang, H. Ma, Y. Zhou, R. Zhang and Z. Song are with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China. (Corresponding author: Hui Ma.) Email: {wangti, mahui, zhouyongbin, r-zhang, songzishuai}@iie.ac.cn.

encapsulates a symmetric-encryption key which could be used to encrypt the data, and the data ciphertext is the DEM part. As far as we know, there is no solution can prevent DDoS attacks on ABE ciphertexts. So the second challenge is *how to prevent DDoS attacks on ABE ciphertexts (KEM part) and also the data ciphertexts (DEM part)?*

Inefficient Resource Consumption Audit. To guarantee the transparency of cloud charge, the data owner should audit the actual consumed resources. The traditional solution is based on the signature, where the data owner has to download every signature for every interaction between the cloud and consumer, then verifies all the signatures one by one. Thus the data owner suffers huge communication overhead and computation overhead. The third challenge is *how to achieve efficient resource consumption audit for the data owner?*

1.1 Our Contribution

Aiming to solve the above challenges, we propose fully accountable ABE (FA-ABE) for the pay-as-you-go model, which can achieve high decryption efficiency for data consumers, prevention of DDoS attacks on the cloud ciphertexts, efficient consumed services audit for the data owner. Our contribution is three-fold:

1) **Improvement of Decryption Efficiency.** In FA-ABE, we for the first time utilize the outsourced decryption technique [9] in the covert adversary model[1] which can properly model the cloud behaviors in the pay-as-you-go scenes. Note that the outsourced decryption technique [9] cannot be adopted directly, because the cloud provider in the covert adversary model is able to do the following cheats: (1) To maintain the reputation, the cloud may create new valid ABE ciphertexts when data loss or data damage occurs. (2) To save the computational resource, the cloud may return random decryption results that look similar to the real ones, e.g., dummy strings of the same length. We take advantage of the verifiable random function (VRF) to check the correctness of decryption results and make sure that the cloud without the necessary information cannot generate a new valid ABE ciphertext. With the help of outsourced decryption and VRF, FA-ABE improves the communication and computation efficiency of decryption in the covert adversary model.
2) **Prevention of DDoS Attacks.** The main reason for the existence of DDoS attacks on ABE ciphertexts is that every data consumer can download any ABE ciphertexts as s/he wishes, regardless of the decryption permission. We observe that the outsourced decryption technique [9] can be used to prevent DDoS attacks on ABE ciphertexts. Specifically, the cloud provider equipped with a blinded transformation key is able to check whether the data consumer has the decryption permission for the requested data or not. Naturally, user authentication should be added to avoid impersonation attacks, so we utilize VRF to authenticate the data consumer's identity. Moreover, in order to perfect the DDoS defense mechanism and enhance economic practicality, we design the limited access control mechanism for authenticated consumers, who are allowed to download the same file for a limited time over a period of time. Therefore, FA-ABE can prevent DDoS attacks on all the ciphertexts and achieve limited access control.
3) **Fast Consumed Services Audit.** We utilize the synchronized aggregate signature to achieve the resource consumption accountability in an efficient way. The cloud provider first collects the resource consumption proofs (signatures) from data consumers during a period time, then generates one aggregate signature as the charge proof at the end of period and sends it to the data owner, who later only needs to verify one signature to audit the consumed services. In this way, the bandwidth consumption and computation efficiency for the data owner are both optimized.

1.2 Related Work

Attribute-based encryption was introduced by Sahai and Waters [16]. Green *et al.* [9] introduced outsourced decryption in ABE systems such that the complex operations of decryption can be offloaded to the cloud server. In order to verify the correctness of outsourced decryption result returned by the cloud, Lai *et al.* [11] formally introduced the verifiability of ABE and proposed a verifiable outsourced decryption ABE (VO-ABE) scheme. Subsequently, a series of work [5], [12]–[14] had been proposed to optimize the efficiency of VO-ABE. We remark that all these schemes consider the cloud as a semi-honest adversary. The verifiability mechanisms can only check the correctness of outsourced decryption result, but cannot detect forged ciphertexts.

For practical application of ABE in the cloud environment, Yuen *et al.* [17] introduced the notion of limited time attribute-based anonymous access control. Ning *et al.* [15] proposed a limited time access control mechanism based on an underlying CP-ABE scheme [18]. Both schemes are suitable to restrict the download time of individual authorized consumer, but not arbitrarily download behaviors of all authorized consumers, which should be considered in the pay-as-you-go model.

Xue *et al.* [4] proposed two well-designed data sharing protocols in the pay-as-you-go model for the first time. The core idea is that the cloud utilizes ABE encryption/decryption game as challenge-response authentication of the data consumer before the download. If the consumer can recover a random challenge value encrypted under the same access policy of the file, then s/he is authorized and the cloud returns the data ciphertext. The work provides an effective solution against DDoS attacks on the DEM, and achieves resource consumption accountability at the same time. However, since the cloud always responds to requests and sends ABE ciphertexts to the consumer, it cannot protect against DDoS attacks on ABE ciphertexts (KEM). The computation cost of ABE decryption for data consumers and overhead of signature verification for data owners are both high.

Another quite similar primitive of accountable ABE is the traceable CP-ABE scheme [19], [20]. Any user and the authority who exactly leak a decryption key to the third user

1. It means the cloud provider may deviate arbitrarily from the protocol specification in an attempt to cheat, but do not wish to be "caught" doing so [10]. "Covert" adversary is stronger than "semi-honest" adversary.

TABLE 1: Function Comparison with Other Related Work*

Schemes	Owner → Consumer	Consumer → Cloud			Cloud → Consumer			Owner → Cloud		Adversary Model of the Cloud	Security Model
	Confidential	Unforged	Outsourced	Verified	KEM	DEM	Limited	RCA	RCAA		
[9], [6], [7]	√	×	√	×	×	×	×	×	×	semi-honest	standard
[11]–[14], [5]	√	×	√	√	×	×	×	×	×	semi-honest	standard
[15]	√	×	√	×†	×‡	×	√	×	×	semi-honest	standard
[4]	√	√	×	×	×	√	×	√	×	covert	random oracle
Ours	√	√	√	√	√	√	√	√	√	covert	standard

* Arrows mean the audit relationship in the system. Confidential stands for data confidentiality, Unforged stands for data unforgeability, Outsourced stands for outsourced decryption, Verified stands for outsourced decryption verifiability, KEM (DEM) stands for the defense of DDoS attacks on the KEM (DEM) part of ciphertexts, Limited stands for limited time access controllability, RCA stands for resource consumption accountability, and RCAA stands for resource consumption aggregate accountability.
† It is defined as auditability. The data consumer can only verify outsourced decryption results with an auditor's help.
‡ The prevention of DDoS attacks is not considered. In this construction, the cloud public key is embedded into data consumers' secret keys. Consumers cannot decrypt ABE ciphertexts without the cloud executing outsourced decryption.

will be identified. Zhang *et al.* [21] proposed a fully-secure white-box traceable CP-ABE scheme supporting large universe and any monotonic access structures, and also applied the scheme in the smart health system [22].

Table 1 compares some functions in our proposed FA-ABE with those in related work. Our construction achieves all-sided accountability and high efficiency.

2 PRELIMINARY

In this section, we review some notations and describe cryptosystems on which our construction is based.

2.1 Notations

$\mathbf{Alg}(x, y, \cdots) \rightarrow z$ denotes that z is obtained by running an algorithm **Alg** on inputs $(x, y, \cdots)$. If x and y are two strings, $|x|$ denotes the length of x, and $x \parallel y$ denotes the concatenation of x and y. For $n \in \mathbb{N}$, $[n] = \{1, 2, \ldots, n\}$. If S is a set, $|S|$ denotes its size and $s \xleftarrow{\$} S$ denotes the operation of selecting an element s uniformly at random from S. $negl(\lambda)$ denotes a negligible function, i.e., $\forall n > 0$, $\exists \lambda_0 \in \mathbb{N}$, s.t., $\lambda > \lambda_0$, $negl(\lambda) < 1/\lambda^n$.

Definition 1 (Prime Order Bilinear Maps)**.** *Let $\mathcal{G}$ be a group generator algorithm that takes as input a security parameter λ and outputs a tuple $D = (p, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups of prime order p, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an efficiently computable bilinear map satisfying that: (1) Bilinearity: $\forall g, h \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_p^*$, we have $e(g^a, h^b) = e(g, h)^{ab}$; (2) Non-degeneracy: $e(g, h) \neq 1$, whenever g, h are not the identity of $\mathbb{G}$.*

Definition 2 (Access Structure [23])**.** *Let $\mathcal{U}$ be the attribute universe. An access structure on $\mathcal{U}$ is a collection $\mathbb{A}$ of non-empty sets of attributes, i.e. $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets. Additionally, an access structure is called monotone for $\forall B, C$, if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.*

Definition 3 (Linear Secret Sharing Schemes (LSSS) [23])**.** *Let p be a prime and $\mathcal{U}$ the attribute universe. A secret sharing scheme Π with domain of secrets $\mathbb{Z}_p$ realizing access structures on $\mathcal{U}$ is linear over $\mathbb{Z}_p$ if:*
(1) The shares of the parties form a vector over $\mathbb{Z}_p$;
(2) For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a matrix $M \in \mathbb{Z}_p^{l \times n}$, called the share-generating matrix, and a function ρ, that labels the rows of M with attributes from $\mathcal{U}$. During the generation of the shares, we consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)^\top$, where $r_2, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_p$. Then the vector of l shares of the secret s according to Π is equal to $M\vec{v} \in \mathbb{Z}_p^{l \times 1}$. The share $(M\vec{v})_i$ belongs to attribute $\rho(i)$.

2.2 CP-AB-KEM with Outsourced Decryption

Define the function $f(\mathcal{S}, \mathbb{A}) = 1$ iff the attribute set $\mathcal{S}$ satisfies the access structure $\mathbb{A}$. The ciphertext policy attribute based key encapsulation mechanism (CP-AB-KEM) with outsourced decryption [9] consists of five algorithms:

- $\mathsf{Setup}(1^\lambda, \mathcal{U}) \rightarrow (PP, MSK)$. On input a security parameter λ and a universe description $\mathcal{U}$, which defines a set of allowed attributes, it outputs public parameters PP and a master secret key MSK.
- $\mathsf{KeyGen}(PP, MSK, \mathcal{S}) \rightarrow (TK, RK)$. On input PP, MSK and a set of attributes $\mathcal{S}$, it outputs a transformation key TK and a retrieval key RK.
- $\mathsf{Encrypt}(PP, \mathbb{A}) \rightarrow (\mathsf{EK}, CT)$. On input PP and an access structure $\mathbb{A}$, it outputs an encapsulated key EK and a ciphertext CT of EK.
- $\mathsf{Transform}(TK, CT) \rightarrow TCT$ or $\perp$. On input TK for an attribute set $\mathcal{S}$ and CT for an access structure $\mathbb{A}$, if $f(\mathcal{S}, \mathbb{A}) = 1$, it outputs a partially decrypted ciphertext (transformed ciphertext) TCT, or outputs $\perp$ otherwise.
- $\mathsf{Decrypt}(RK, TCT) \rightarrow \mathsf{EK}$. On input RK and TCT, it outputs EK.

We utilize the CP-AB-KEM with outsourced decryption to achieve the prevention of DDoS attacks on ABE ciphertexts, and optimize the efficiency for data consumers: (1) The cloud can use the transformation key of a consumer to verify his/her decryption permission before the download. If the attribute set in the transformation key does not satisfy the access structure of the request ciphertext, the cloud will not execute outsourced decryption and respond nothing. (2) With outsourced decryption, the communication cost of ciphertexts and computation cost of decryption for data consumers are both reduced to constant scale.

We require the CP-AB-KEM with outsourced decryption is (selectively) indistinguishable under chosen-plaintext attack (IND-CPA) secure. Please refer to [18] for more details.

2.3 Synchronized Aggregate Signature

The synchronized aggregate signature (SAS) consists of six algorithms:

- $\mathsf{Setup}(1^\lambda) \rightarrow (PP)$. On input a security parameter λ, it outputs public parameters PP.
- $\mathsf{KeyGen}(PP) \rightarrow (pk, sk)$. On input PP, it outputs a public key pk and a private key sk.
- $\mathsf{Sign}(PP, sk, m, \omega) \rightarrow \sigma$. On input PP, sk, a message m and a time period ω, it outputs a signature σ.
- $\mathsf{Verify}(PP, pk, m, \sigma) \rightarrow 1$ or 0. On input PP and a signature σ on a message m under a public key pk, it outputs 1 or 0 depending on the validity of σ.
- $\mathsf{Agg}(PP, \mathbf{PK}, \mathbf{Msg}, \mathbf{Sig}) \rightarrow \sigma_\Sigma$. On input PP, individual signatures $\mathbf{Sig} = (\sigma_1, \ldots, \sigma_l)$ on messages $\mathbf{Msg} = (m_1, \ldots, m_l)$ under public keys $\mathbf{PK} = (pk_1, \ldots, pk_l)$, it outputs an aggregate signature σ_Σ.
- $\mathsf{AggVerify}(PP, \mathbf{PK}, \mathbf{Msg}, \sigma_\Sigma) \rightarrow 1$ or 0. On input PP, an aggregate signature σ_Σ on messages $\mathbf{Msg} = (m_1, \ldots, m_l)$ under public keys $\mathbf{PK} = (pk_1, \ldots, pk_l)$, it outputs 1 or 0 depending on the validity of σ_Σ.

We utilize the SAS to realize efficient resource consumption audit, and achieve the limited time access control mechanism: (1) The synchronized aggregate signature can aggregate all signatures during the same time period into a short aggregate signature. Thus there is no need to transmit massive signatures (resource consumption proofs) to the data owner, who only needs to verify one signature to audit the consumed resources. (2) During the encryption phase in our construction, the data owner prepares κ signing key pairs for the upload file, where κ is the download limited time in one period. Each signing key can be used to generate a signature (proof) at most once for each period, otherwise the aggregate signature is invalid. Therefore, excessive downloads from all authorized consumers can be limited.

We require the SAS is existentially unforgeable under chosen message attack (EUF-CMA) secure. Please refer to [24] for more details.

2.4 Verifiable Random Function

Verifiable Random Functions (VRFs) as introduced by Micali, Rabin and Vadhan [25] are a special form of Pseudo Random Functions (PRFs). A secret key holder in addition to computing the functions value y at any point x, can also generate a non-interactive proof π that y is correct, without compromising pseudorandomness at other points [26]. A function family $F_{(\cdot)}(\cdot) : \{0,1\}^m \rightarrow \{0,1\}^n$ is a family of VRFs consisting of four algorithms:

- $\mathsf{Setup}(1^\lambda) \rightarrow PP$. On input a security parameter λ, it outputs public parameters PP.
- $\mathsf{KeyGen}(PP) \rightarrow (PK, SK)$. On input PP, it outputs a public key PK and a private key SK.
- $\mathsf{Prove}(PP, SK, x) \rightarrow (y, \pi)$. On input PP, SK and an input x, it outputs $(y, \pi) = (F_{SK}(x), \pi_{SK}(x))$, where y is the VRF evaluation and π is the proof of correctness.
- $\mathsf{Verify}(PP, PK, x, y, \pi) \rightarrow 1$ or 0. It verifies whether y is the correct evaluation on input x.

Formally, the security of VRF is as follows:

1) **Uniqueness:** no values $(PP, PK, x, y_1, y_2, \pi_1, \pi_2)$ can satisfy $\mathsf{Verify}(PP, PK, x, y_1, \pi_1) = \mathsf{Verify}(PP, PK, x, y_2, \pi_2) = 1$ when $y_1 \neq y_2$.
2) **Provability:** if $(y, \pi) = \mathsf{Prove}(PP, SK, x)$, then $\mathsf{Verify}(PP, PK, x, y, \pi) = 1$.
3) **Pseudorandomness:** for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, who does not query its oracle on x, the following advantage:

$$\left| \Pr \left[b' = b \left| \begin{array}{r} (PK, SK) \leftarrow \mathsf{KeyGen}(PP); \\ (x) \leftarrow \mathcal{A}^{\mathsf{Prove}(\cdot)}(PK); \\ y_0 = F_{SK}(x); y_1 \xleftarrow{\$} \{0,1\}^n; \\ b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}^{\mathsf{Prove}(\cdot)}(y_b) \end{array} \right. \right] - 1/2 \right| \leq negl(\lambda).$$

We utilize the provability of VRF to realize authentication that each data consumer needs to generate a request with his/her secret key before the download. In this way, impersonation attacks on the cloud provider can be avoided.

Moreover, taking advantage of VRF, our construction achieves both data unforgeability and outsourced decryption verifiability simultaneously: (1) Data owner uses his/her secret key to compute a pair of verification values (y, π) for the upload file. The pseudorandomness of VRF guarantees that without the secret key, the cloud cannot forge such values. (2) To verify the correctness of outsourced decryption results, (y, π) should determine the original file uniquely. Thus we require the VRF is an injective function, e.g., the constructions of [27], [28]. Specifically, it satisfies:
Binding: no values (SK, x_1, x_2) can satisfy $F_{SK}(x_1) = F_{SK}(x_2)$ when $x_1 \neq x_2$.

Note that in [28], $F_{SK}(x) = e(g,g)^{1/(x+SK)}$ is an injective function, in which an element from the group $\mathbb{Z}_p$ with the prime order p is mapped to an element from the group $\mathbb{G}_T$ with the same prime order p. With a fixed SK, the exponent $1/(x + SK)$ uniquely determines the input $x \in \mathbb{Z}_p$. If we use a collision-resistant hash function $\mathrm{H}(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^m$ to define VRF with unrestricted input length, the composition $F_{(\cdot)}(\mathrm{H}(\cdot)) : \{0,1\}^* \rightarrow \{0,1\}^n$ is trivially secure, and the binding property is preserved by the collision resistance of hash function. We use the VRF proposed in [28] to instantiate our construction.

3 System Model and Security Model

In this section, we present the system model, adversarial model and security requirements of FA-ABE.

3.1 System Model

As illustrated in Fig. 1, four entities are involved in our system, namely the ***private key generator (PKG)***, the ***cloud provider (Cloud)***, the ***data owner (Owner)*** and the ***data consumer (Consumer)***. The data owner can also be a data consumer in the system, and vice versa. The acronyms used in this paper are given in Table 2.

- ***Private Key Generator*** is responsible to set up the system parameters and distributes all the cryptographic keys to other entities.
- ***Data Owner*** hires the cloud provider and pays for resource consumption periodically according to the proof. The data owner encrypts files associated with needed access policies and download frequencies before uploading them to the cloud provider.
- ***Data Consumer*** wants to download any encrypted files of his/her interest from the cloud. If the received ciphertext is not forged and calculated correctly, the

TABLE 2: Acronyms Used in This Paper

Acronym	Description
PKG	private key generator
PP	public parameters
MSK	master secret key
(PK_o, SK_o)	data owner's verification key pair
(PK_u, SK_u)	data consumer's authentication key pair
TK_u	data consumer's transformation key
RK_u	data consumer's retrieval key
CT	ciphertext of the file
TCT	transformed ciphertext of the file
CK	set of κ signing key pairs
CSK_j	ciphertext of the j^{th} signing secret key
ctr	download counter
Info	information of the request file
req	download request
proof	resource consumption proof
proof$_\Sigma$	resource consumption aggregate proof

data consumer generates a resource consumption proof for this service.

- ***Cloud Provider*** is deployed to offer data storage and outsourced decryption services. It should reject download requests from unauthorized consumers and excessive requests from authorized consumers. It also collects resource consumption proofs and generates an aggregate proof to charge the data owner.

Next, we give an overview of the system process. There are four phases in FA-ABE:

1) Setup and Register Phase. The PKG runs **Setup.pkg**() to generate public parameters and the master secret key. For the data owner, the PKG runs **KeyGen.pkg**() to generate a verification key pair (PK_o, SK_o) and publishes PK_o. For the data consumer, the PKG runs **KeyGen.pkg**() to generate an authentication key pair (PK_u, SK_u), a transformation key TK_u and a retrieval key RK_u. (PK_u, SK_u, RK_u) are transmitted to the consumer, and (PK_u, TK_u) are transmitted to the cloud provider.

2) Encrypt and Upload Phase. Before uploading the file to the cloud, the data owner sets the access policy and runs **Encrypt.owner**() to encrypt the file. In order to control the download frequency (the limited time in one period) and achieve the resource consumption accountability, the data owner provides several signing keys during the encryption, where the secret keys are encrypted under the same access policy of the file. The file ciphertext CT and the signing keys set CK are uploaded to the cloud provider.

3) Download and Decrypt Phase. The data consumer runs **Request.consumer**() to generate a download request **req** and sends it to the cloud provider, who later runs **DecControl.cloud**() to verify it. If **req** is valid, the cloud executes outsourced decryption and returns the transformed ciphertext TCT and a secret key ciphertext CSK_{ctr} to the data consumer. If **req** is invalid, the request is rejected. The consumer runs **DecVerify.consumer**() to decrypt and verify the received TCT and CSK_{ctr}.

4) Prove and Verify Phase. If TCT is not forged and calculated correctly, the data consumer runs **Prove.consumer**() to generate a resource consumption proof **proof** with the signing secret key encapsulated in CSK_{ctr}, and sends it to the cloud provider. At the end of each period, the cloud runs **ProofAgg.cloud**() to aggregate all proofs in the period into **proof**$_\Sigma$ for the data owner, who runs **ProofVerify.owner**() to verify the correctness of **proof**$_\Sigma$ and pays for the charge.

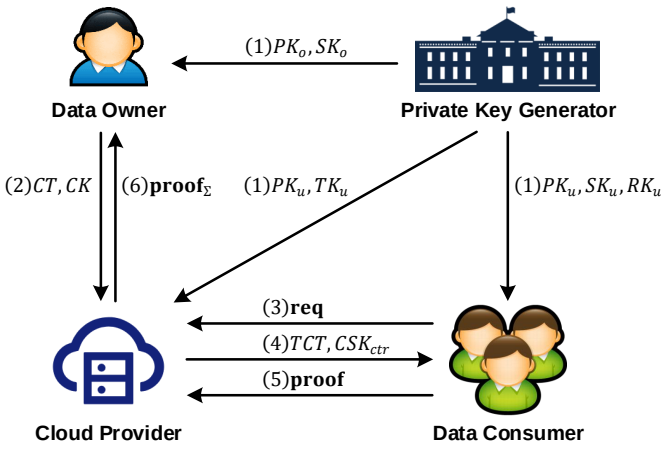


Fig. 1: System Model of FA-ABE

3.2 Adversarial Model

In FA-ABE, the PKG and data owners are assumed to be fully trusted. We consider the following two types of adversaries: data consumers and the cloud provider.

(1) Data consumers may have these behaviors:

- Unauthorized consumers may collude together and try to decrypt a ciphertext that is beyond their decryption permissions. Besides, they may frequently request ABE ciphertexts (KEM part) or data ciphertexts (DEM part) to raise DDoS attacks.
- Authorized consumers who have the decryption permission of a file may arbitrarily download the file in a short period to raise DDoS attacks.

(2) The cloud provider is considered to be a "covert" adversary [10], which is stronger than a semi-honest adversary[2], but weaker than a malicious one[3]. More specifically, covert adversaries may deviate arbitrarily from the protocol specification in an attempt to cheat, but do not wish to be "caught" doing so. A deterrence factor $0 < \epsilon \leq 1$ is usually set in the system. Any attempt to cheat by an adversary can be detected by the honest parties with probability at least ϵ. We design an efficient mechanism with $\epsilon = 1$, thus the cheating cloud provider will always be caught so refrain from attempting to cheat. The cloud provider may have the following behaviors:

- The cloud may lose or damage the original data for some reason, such as storage system corruption. In order to maintain the provider reputation, the cloud may try to create new valid ciphertexts.
- The cloud may become "lazy" in order to save computation power, as long as it cannot be detected. Namely,

2. Semi-honest (honest-but-curious) adversaries correctly follow the protocol specification, yet may attempt to learn additional information by analyzing the transcript of messages received during the execution.

3. Malicious adversaries may behave arbitrarily and are not bound in any way to following the instructions of the specified protocol.

the cloud does not execute the outsourced decryption but just returns the data consumer a random result which is indistinguishable with a real one.
- The cloud may try to forge the resource consumption proof to charge the data owner more.

3.3 Security Requirements

Based on the above adversarial model, we have five security requirements in FA-ABE. The formal definitions are given in Appendix A.

- ***Data Confidentiality.*** Unauthorized consumers and the cloud provider cannot obtain any information from the ciphertext. Namely, if none of their attribute sets satisfies the access policy of a ciphertext, when they collude together by combining their attributes, they cannot decrypt the ciphertext.
- ***Data Download Controllability.*** Download behaviors are strictly controlled by the cloud provider, and the DDoS attacks from both unauthorized consumers and authorized consumers are all prevented. Namely, unauthorized consumers cannot get any data from the cloud; all authorized consumers in each period can only download the same file with limited times.
- ***Data Unforgeability.*** The data stored on the cloud is integrated and authenticated. Namely, data consumers can verify the data is indeed created by the data owner, rather than forged by the cloud provider.
- ***Outsourced Decryption Verifiability.*** When the cloud provider executes outsourced decryption incorrectly, the behavior will be detected by the data consumer.
- ***Resource Consumption Accountability.*** The cloud provider cannot forge a valid resource consumption proof to cheat the data owner for more charge.

4 Fully Accountable ABE

In this section, we present the generic construction of fully accountable ABE (FA-ABE), and analyze the security.

4.1 Details of Construction

Our generic construction of FA-ABE has the following building blocks: a CP-AB-KEM with outsourced decryption, a symmetric-key encryption, a synchronized aggregate signature and a verifiable random function.

1) Setup and Register Phase (SR): As illustrated in Fig. 2, all system entities implement this operation, which consists of the following three steps:

- ***SR-Step1.*** The PKG generates system parameters PP and the muster secret key MSK.

$$\mathsf{ABE.Setup}(1^\lambda, \mathcal{U}) \to (PP_{\mathrm{ABE}}, MSK),$$
$$\mathsf{VRF.Setup}(1^\lambda) \to PP_{\mathrm{VRF}},$$
$$\mathsf{SAS.Setup}(1^\lambda) \to PP_{\mathrm{SAS}}.$$

It chooses a key derivation function KDF with the output length $\mathcal{L}$. The public parameters are

$$PP = (PP_{\mathrm{ABE}}, PP_{\mathrm{VRF}}, PP_{\mathrm{SAS}}, \mathsf{KDF}, \mathcal{L}).$$

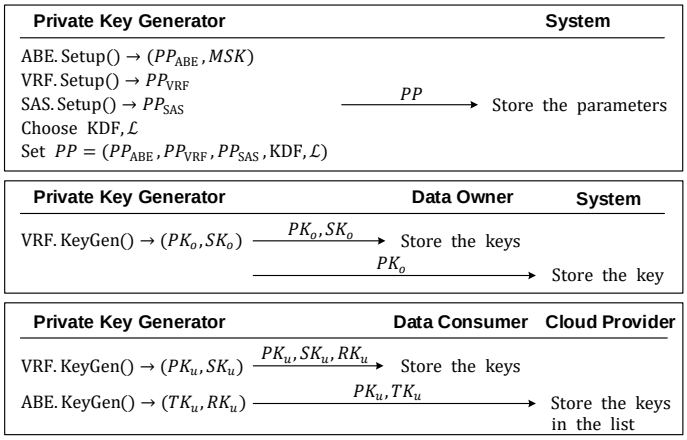


Fig. 2: Setup and Register Phase

- ***SR-Step2.*** For the data owner, the PKG generates a verification key pair (PK_o, SK_o).

$$\mathsf{VRF.KeyGen}(PP_{\mathrm{VRF}}) \to (PK_o, SK_o).$$

PK_o is published to other entities.
- ***SR-Step3.*** When a data consumer joins the system, the PKG generates an authentication key pair (PK_u, SK_u). According to the attribute set $\mathcal{S}$ of the consumer, the PKG also generates a transformation key TK_u and a retrieval key RK_u.

$$\mathsf{VRF.KeyGen}(PP_{\mathrm{VRF}}) \to (PK_u, SK_u),$$
$$\mathsf{ABE.KeyGen}(PP_{\mathrm{ABE}}, MSK, \mathcal{S}) \to (TK_u, RK_u).$$

(PK_u, SK_u, RK_u) are sent to the data consumer, and (PK_u, TK_u) are sent to the cloud provider, who maintains an authentication list for all consumers.

2) Encrypt and Upload Phase (EU): As illustrated in Fig. 3, the data owner and the cloud provider implement this operation, which consists of the following three steps:

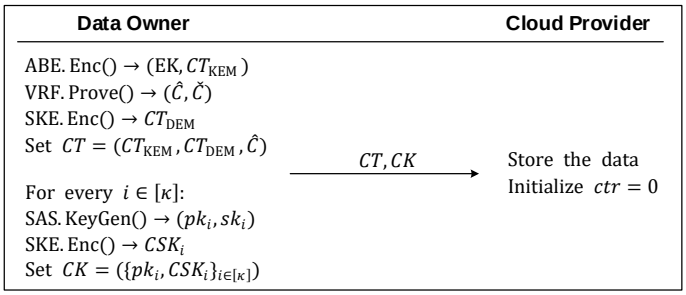


Fig. 3: Encrypt and Upload Phase

- ***EU-Step1.*** The data owner generates an encapsulated key EK under required access policy, which is formulated as an access structure $\mathbb{A}$ over attributes.

$$\mathsf{ABE.Enc}(PP_{\mathrm{ABE}}, \mathbb{A}) \to (\mathsf{EK}, CT_{\mathrm{KEM}}).$$

- ***EU-Step2.*** The data owner utilizes the secret key SK_o to generate a pair of verification values $(\hat{C}, \check{C})$, then de-

rives a symmetric key from EK, and encrypts the file m using the symmetric-key encryption (SKE) algorithm.

$$\mathsf{VRF.Prove}(PP_{\text{VRF}}, SK_o, m \parallel \mathsf{EK}) \rightarrow (\hat{C}, \check{C}),$$
$$\mathsf{SKE.Enc}(\mathsf{KDF}(\mathsf{EK}, \mathcal{L}), m \parallel \check{C}) \rightarrow CT_{\text{DEM}},$$

where $\hat{C}$ is the VRF evaluation and $\check{C}$ is the proof of correctness.

- ***EU-Step3.*** The data owner defines the file's download frequency, i.e., the limited time κ in a particular period, such as 10 times per day. The owner generates κ signing key pairs, and encrypts secret keys using the same symmetric key derived from EK. For every $i \in [\kappa]$,

$$\mathsf{SAS.KeyGen}(PP_{\text{SAS}}) \rightarrow (pk_i, sk_i),$$
$$\mathsf{SKE.Enc}(\mathsf{KDF}(\mathsf{EK}, \mathcal{L}), sk_i) \rightarrow CSK_i.$$

The ciphertexts $CT = (CT_{\text{KEM}}, CT_{\text{DEM}}, \hat{C})$ and $CK = (\{pk_i, CSK_i\}_{i\in[\kappa]})$ are uploaded to the cloud, who initializes the download counter of CT with $ctr = 0$.

3) Download and Decrypt Phase (DD): As illustrated in Fig. 4, the data consumer and the cloud provider interactively implement this operation, which consists of the following four steps:

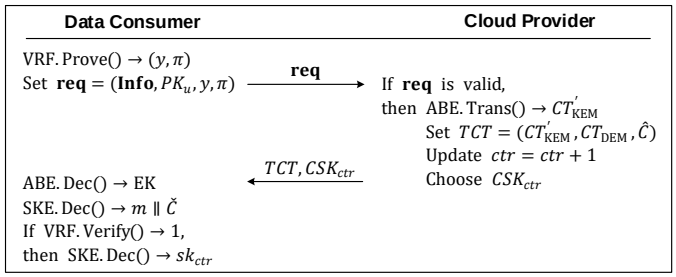


Fig. 4: Download and Decrypt Phase

- ***DD-Step1.*** The data consumer utilizes SK_u to generate a download request.

$$\mathsf{VRF.Prove}(PP_{\text{VRF}}, SK_u, \mathbf{Info}) \rightarrow (y, \pi),$$

where $\mathbf{Info} = (\text{filename}, \text{timestamp})$. The request $\mathbf{req} = (\mathbf{Info}, PK_u, y, \pi)$ is sent to the cloud.
- ***DD-Step2.*** The cloud provider searches the key term (PK_u, TK_u) in the authentication list according to PK_u in the request $\mathbf{req}$, and verifies the request. The request is valid if all the following conditions hold.

$$\mathsf{VRF.Verify}(PP_{\text{VRF}}, PK_u, \mathbf{Info}, y, \pi) \rightarrow 1,$$
$$f(\mathcal{S}, \mathbb{A}) = 1, \quad \text{and} \quad ctr < \kappa,$$

where $\mathcal{S}$ is the attribute set of TK_u, $\mathbb{A}$ is the access policy of the request ciphertext CT, ctr is the download counter of CT in the current period. If $\mathbf{req}$ is valid, the cloud executes outsourced decryption on CT_{KEM}.

$$\mathsf{ABE.Trans}(TK_u, CT_{\text{KEM}}) \rightarrow CT'_{\text{KEM}}.$$

The cloud updates the counter $ctr = ctr+1$, and returns the transformed ciphertext $TCT = (CT'_{\text{KEM}}, CT_{\text{DEM}}, \hat{C})$ and the secret key ciphertext CSK_{ctr} to the consumer.

- ***DD-Step3.*** The data consumer decrypts the transformed ABE ciphertext CT'_{KEM} to recover EK, and then decrypts the data ciphertext CT_{DEM}.

$$\mathsf{ABE.Dec}(RK_u, CT'_{\text{KEM}}) \rightarrow \mathsf{EK},$$
$$\mathsf{SKE.Dec}(\text{KDF}(\mathsf{EK}, \mathcal{L}), CT_{\text{DEM}}) \rightarrow (m, \check{C}).$$

- ***DD-Step4.*** The data consumer utilizes the data owner's public key PK_o to verify whether the received TCT is forged or calculated incorrectly. If TCT is valid, the consumer recovers the signing secret key sk_{ctr}.

$$\text{If} \quad \mathsf{VRF.Verify}(PP_{\text{VRF}}, PK_o, m \parallel \mathsf{EK}, \hat{C}, \check{C}) \rightarrow 1,$$
$$\text{then} \quad \mathsf{SKE.Dec}(\text{KDF}(\mathsf{EK}, \mathcal{L}), CSK_{ctr}) \rightarrow sk_{ctr}.$$

4) Prove and Verify Phase (PV): As illustrated in Fig. 5, the data consumer, the cloud provider and the data owner implement this operation, which consists of the following three steps:

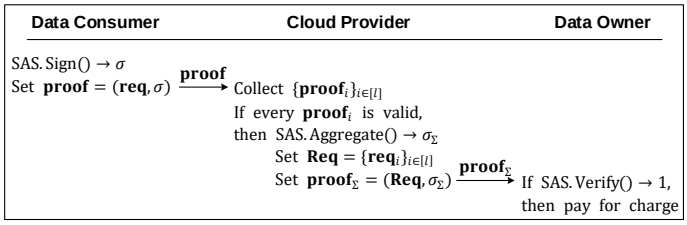


Fig. 5: Prove and Verify Phase

- ***PV-Step1.*** The data consumer uses the signing secret key sk_{ctr} to generate a signature.

$$\mathsf{SAS.Sign}(PP_{\text{SAS}}, sk_{ctr}, \mathbf{req}) \rightarrow \sigma.$$

We assume the consumer will faithfully generate the signature if TCT is valid. The resource consumption proof $\mathbf{proof} = (\mathbf{req}, \sigma)$ is transmitted to the cloud.
- ***PV-Step2.*** The cloud provider collects all resource consumption proofs $\mathbf{Proof} = (\mathbf{proof}_1, \ldots, \mathbf{proof}_l)$ during each period. At the end of each period, the cloud first uses the corresponding public key pk_i in CK to verify $\mathbf{proof}_i = (\mathbf{req}_i, \sigma_i)$. The purpose of verification is to make sure each proof is valid. If for $i \in [l]$,

$$\mathsf{SAS.Verify}(PP_{\text{SAS}}, pk_i, \mathbf{req}_i, \sigma_i) \rightarrow 1,$$

then the cloud aggregates all proofs into an aggregate proof $\mathbf{proof}_\Sigma$. As long as the proofs are generated in the same period (the same date or the same hour), which could be generated for download services of different files, they can be aggregated into a short proof.

$$\mathsf{SAS.Agg}(PP_{\text{SAS}}, \mathbf{PK}, \mathbf{Req}, \mathbf{Sig}) \rightarrow \sigma_\Sigma,$$

where $\mathbf{PK} = (pk_1, \ldots, pk_l)$, $\mathbf{Req} = (\mathbf{req}_1, \ldots, \mathbf{req}_l)$ and $\mathbf{Sig} = (\sigma_1, \ldots, \sigma_l)$. The aggregate proof $\mathbf{proof}_\Sigma = (\mathbf{Req}, \sigma_\Sigma)$ is transmitted to the data owner. All download counters are reset at the beginning of next period.
- ***PV-Step3.*** The data owner verifies the validity of $\mathbf{proof}_\Sigma$, and pays for the cloud charge if:

$$\mathsf{SAS.AggVerify}(PP_{\text{SAS}}, \mathbf{PK}, \mathbf{Req}, \sigma_\Sigma) \rightarrow 1.$$

4.2 Security Analysis

In this section, we analyze the construction on how to achieve the security requirements.

Theorem 1 (Data Confidentiality)**.** *Suppose that the CP-ABE-KEM with outsourced decryption is (selectively) IND-CPA secure, the symmetric-key encryption algorithm is semantically secure, the* KDF *is secure, and the VRF has the pseudorandomness property, then the constructed* FA-ABE *meets the data confidentiality.*

Proof Outline. The file is encrypted in the hybrid encryption (the KEM/DEM setting) [8], where the CP-AB-KEM is used to encapsulate a symmetric key, and the symmetric-key encryption is used to encrypt the file. The difference between FA-ABE and the standard hybrid encryption is that two verification values $(\hat{C}, \check{C})$ are generated in FA-ABE. Here, we show the modification does not compromise security.

(1) The first verification value $\hat{C}$ is exposed in the ciphertext CT, but it is the output of the verifiable random function. The pseudorandomness of VRF guarantees that the function evaluation is indistinguishable with a random value chosen from the output space. Therefore, $\hat{C}$ does not leak any information about the file m. (2) The second verification value $\check{C}$ is encrypted in the DEM part. Since the hybrid encryption algorithms and the key derivation function are secure, the confidentiality of $\check{C}$ can be guaranteed. Thus the constructed FA-ABE meets the data confidentiality. We postpone the detailed theoretical proof to Appendix B.1.

Theorem 2 (Data Download Controllability)**.** *Suppose that the VRF has the pseudorandomness property, and the synchronized aggregate signature is EUF-CMA secure, then the constructed* FA-ABE *meets the data download controllability.*

Proof Outline. First, we consider the condition of unauthorized data consumers who pretend to be authorized ones. The pseudorandomness of VRF, defined in the form of a "decisional" hard problem, naturally implies a weaker "computational" property that any PPT adversary without the knowledge of SK_u cannot output a tuple (x, y, π) satisfying $y = F_{SK_u}(x)$ and $\pi = \pi_{SK_u}(x)$. Thus unauthorized data consumers cannot generate valid download requests.

Second, for authorized data consumers whose attributes satisfy the access policy, when the download time in the current period reaches the limit, the cloud provider will also refuse their requests. Note that $\mathsf{SAS.AggVerify}()$ outputs 0 if any signing key is used more than once in an aggregate signature. Even though the covert cloud may deviate arbitrarily from the protocol, providing download services for a file more than limited times in one period, it cannot charge the data owner for these services. Thus the constructed FA-ABE meets the data download controllability.

Theorem 3 (Data Unforgeability)**.** *Suppose that the VRF has the property of pseudorandomness, then the constructed* FA-ABE *meets the data unforgeability.*

The proof is very similar with that of data download controllability in the condition of unauthorized data consumers. We postpone the detailed theoretical proof to Appendix B.2.

Theorem 4 (Outsourced Decryption Verifiability)**.** *Suppose that the VRF has the binding property, then the constructed* FA-ABE *meets the outsourced decryption verifiability.*

Proof Outline. The verification value $\hat{C}$ in the ciphertext CT is used to realize the verifiability of outsourced decryption. During the encryption, the data owner using the secret key SK_o to calculate the VRF evaluation $\hat{C}$ on the combination $m \parallel \mathsf{EK}$. If the cloud executes the outsourced decryption incorrectly, namely it sends the data consumer an error transformed ABE ciphertext $CT^*_{\text{KEM}} \neq CT'_{\text{KEM}}$, then the consumer will recover a different combination from CT_{DEM}, i.e., $m^* \parallel \mathsf{EK}^* \neq m \parallel \mathsf{EK}$. Note that the binding of VRF guarantees that with different input, the VRF evaluation is different. Thus $\hat{C}$ does not match $m^* \parallel \mathsf{EK}^*$, and the outsourced decryption result is incorrect. We postpone the detailed theoretical proof to Appendix B.3.

Theorem 5 (Resource Consumption Accountability)**.** *Suppose that the constructed* FA-ABE *meets the data confidentiality, and the synchronized aggregate signature is EUF-CMA secure, then the constructed* FA-ABE *meets the resource consumption accountability.*

Proof Outline. The data confidentiality of FA-ABE guarantees that any PPT adversary cannot get any signing secret key from the set CK. Since the synchronized aggregate signature is EUF-CMA secure, with the knowledge of many proofs, any PPT adversary cannot create a valid aggregate proof containing forged ones.

5 Our Instantiation

In this section, we present an instantiation of FA-ABE with the CP-ABE-KEM [18], the AES algorithm [29], the VRF [28], the synchronized aggregate signature [24], and the KDF [30]. The concrete FA-ABE construction is as follows.

- **Setup.pkg**$(1^\lambda, \mathcal{U}) \to (PP, MSK)$.
  - Call $\mathcal{G}(1^\lambda)$ to get $D = (p, \mathbb{G}, \mathbb{G}_T, e)$, where p is the prime order of the groups $\mathbb{G}$ and $\mathbb{G}_T$.
  - $\mathsf{ABE.Setup}(1^\lambda, \mathcal{U})$: set the attribute universe $\mathcal{U} = \mathbb{Z}_p$, choose $g_1, h, u, v, w \xleftarrow{\$} \mathbb{G}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and set $PP_{\text{ABE}} = (g_1, h, u, v, w, e(g_1, g_1)^\alpha)$.
  - $\mathsf{VRF.Setup}(1^\lambda)$: choose $g_2 \xleftarrow{\$} \mathbb{G}$ and set $PP_{\text{VRF}} = g_2$.
  - $\mathsf{SAS.Setup}(1^\lambda)$: choose $g_3 \xleftarrow{\$} \mathbb{G}$ and set $PP_{\text{SAS}} = g_3$.
  - Adopt the key derivation function KDF, of which the output length $\mathcal{L}$ is equal to the key size of the AES algorithm. Choose two collision-resistant hash functions, $\mathrm{H}_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ and $\mathrm{H}_2 : \{0,1\}^* \to \mathbb{G}$.
  - Output $PP = (PP_{\text{ABE}}, PP_{\text{VRF}}, PP_{\text{SAS}}, D, \mathsf{KDF}, \mathcal{L}, \mathrm{H}_1, \mathrm{H}_2)$ and $MSK = \alpha$.
- **KeyGen.pkg**$(PP, MSK, \mathcal{S}) \to (PK_o, SK_o)$ and (PK_u, SK_u, TK_u, RK_u).
  - $\mathsf{VRF.KeyGen}(PP_{\text{VRF}})$: choose $\gamma \xleftarrow{\$} \mathbb{Z}_p$ and set $(PK_o, SK_o) = (g_2^\gamma, \gamma)$ for the owner, choose $\delta \xleftarrow{\$} \mathbb{Z}_p$ and set $(PK_u, SK_u) = (g_2^\delta, \delta)$ for the consumer.
  - $\mathsf{ABE.KeyGen}(PP_{\text{ABE}}, MSK, \mathcal{S})$: let attribute set $\mathcal{S} = \{A_1, A_2, \ldots, A_k\} \subseteq \mathbb{Z}_p$. Choose $\beta, r, r_1, \ldots, r_k \xleftarrow{\$} \mathbb{Z}_p$, compute $K_0 = g_1^{\alpha/\beta} w^{r/\beta}$, $K_1 = g_1^{r/\beta}$, and for $j \in [k]$:

$$K_{j,2} = g_1^{r_j/\beta}, K_{j,3} = (u^{A_j} h)^{r_j/\beta} v^{-r/\beta}.$$

    Set $TK_u = (\mathcal{S}, K_0, K_1, \{K_{j,2}, K_{j,3}\}_{j\in[k]})$, $RK_u = \beta$.
- **Encrypt.owner**$(PP, SK_o, m, \kappa, (M, \rho)) \to (CT, CK)$.

- $\mathsf{ABE.Enc}(PP_{\mathrm{ABE}}, \mathbb{A})$: let (M, ρ) be an LSSS access structure with $M \in \mathbb{Z}_p^{l\times n}$ and $\rho : [l] \to \mathbb{Z}_p$. Choose $\vec{x} = (s, x_2, \ldots, x_n)^\top \xleftarrow{\$} \mathbb{Z}_p$. The vector of the shares of s is $\vec{\lambda} = (\lambda_1, \ldots, \lambda_l)^\top = M\vec{x}$. Choose $t_1, t_2, \ldots, t_l \xleftarrow{\$} \mathbb{Z}_p$, compute $C_0 = g_1^s$, and for $i \in [l]$:

  $$C_{i,1} = w^{\lambda_i} v^{t_i}, C_{i,2} = (u^{\rho(i)}h)^{-t_i}, C_{i,3} = g_1^{t_i}.$$

  Set $CT_{\mathrm{KEM}} = ((M, \rho), C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i\in[l]})$, and $\mathsf{EK} = e(g_1, g_1)^{\alpha s}$.
  - $\mathsf{VRF.Prove}(PP_{\mathrm{VRF}}, SK_o, m \parallel \mathsf{EK})$: calculate $\hat{C} = e(g_2, g_2)^{1/(\mathrm{H}_1(m\parallel \mathsf{EK})+SK_o)}$, $\check{C} = g_2^{1/(\mathrm{H}_1(m\parallel \mathsf{EK})+SK_o)}$.
  - $\mathsf{SKE.Enc}()$: $CT_{\mathrm{DEM}} = \mathrm{AES.Enc}(\mathrm{KDF}(\mathsf{EK}, \mathcal{L}), m \,\|\, \check{C})$.
  - Suppose the file can be downloaded at most κ times in one period. For $i \in [\kappa]$:
    $\mathsf{SAS.KeyGen}(PP_{\mathrm{SAS}})$: choose $sk_i = x_i \xleftarrow{\$} \mathbb{Z}_p$ and set $pk_i = X_i = g_3^{x_i}$.
    $\mathsf{SKE.Enc}()$: $CSK_i = \mathrm{AES.Enc}(\mathrm{KDF}(\mathsf{EK}, \mathcal{L}), sk_i)$.
  - Output $CT = \{CT_{\mathrm{KEM}}, CT_{\mathrm{DEM}}, \hat{C}\}$ and $CK = \{pk_i, CSK_i\}_{i\in[\kappa]}$.
- **Request.consumer**$(PP, PK_u, SK_u, \mathbf{Info}) \to \mathbf{req}$.
  - $\mathsf{VRF.Prove}(PP_{\mathrm{VRF}}, SK_u, \mathbf{Info})$: set the request information as **Info**, calculate $y = e(g_2, g_2)^{1/(\mathrm{H}_1(\mathbf{Info})+SK_u)}$, $\pi = g_2^{1/(\mathrm{H}_1(\mathbf{Info})+SK_u)}$.
  - Output $\mathbf{req} = (\mathbf{Info}, PK_u, y, \pi)$.
- **DecControl.cloud**$(PP, \mathbf{req}, TK_u, CT) \to TCT$ or $\perp$.
  - Halt if one of the following conditions holds:
    (1) $\mathsf{VRF.Verify}(PP_{\mathrm{VRF}}, PK_u, \mathbf{info}, y, \pi)$: $y \neq e(g_2, \pi)$, or $e(g_2^{\mathrm{H}_1(\mathbf{Info})} \cdot PK_u, \pi) \neq e(g_2, g_2)$;
    (2) the attribute set of TK_u does not satisfy the access structure of CT;
    (3) the download time of CT in the current period is κ.
  - $\mathsf{ABE.Trans}(TK_u, CT_{\mathrm{KEM}})$: calculate $I = \{i : \rho(i) \in \mathcal{S}\}$ and the constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} \omega_i \vec{M}_i = (1, 0, \ldots, 0)$, where $\vec{M}_i$ is the i-th row of the matrix M. Then compute

    $$CT'_{\mathrm{KEM}} = \frac{e(C_0, K_0)}{\prod_{i\in I}(e(C_{i,1}, K_1)e(C_{i,2}, K_{j,2})e(C_{i,3}, K_{j,3}))^{\omega_j}} = e(g_1, g_1)^{\alpha s/\beta}$$

    where j is the index of the attribute $\rho(i)$ in $\mathcal{S}$.
  - Update the counter $ctr = ctr + 1$, and output $TCT = (CT'_{\mathrm{KEM}}, CT_{\mathrm{DEM}}, \hat{C})$.
- **DecVerify.consumer**$(PP, PK_o, RK_u, TCT, CSK_{ctr}) \to (m, sk_i)$ or $\perp$.
  - $\mathsf{ABE.Dec}(RK_u, CT'_{\mathrm{KEM}})$: $\mathsf{EK} = CT'^{\beta}_{\mathrm{KEM}} = e(g_1, g_1)^{\alpha s}$.
  - $\mathsf{SKE.Dec}()$: $m \,\|\, \check{C} = \mathrm{ABE.Dec}(\mathrm{KDF}(\mathsf{EK}, \mathcal{L}), CT_{\mathrm{DEM}})$.
  - $\mathsf{VRF.Verify}(PP_{\mathrm{VRF}}, PK_o, m \,\|\, \mathsf{EK}, \hat{C}, \check{C})$: if $\hat{C} \neq e(g_2, \check{C})$ or $e(g_2^{\mathrm{H}_1(m\parallel \mathsf{EK})} \cdot PK_o, \check{C}) \neq e(g_2, g_2)$, halt.
  - Compute $sk_{ctr} = \mathrm{ABE.Dec}(\mathrm{KDF}(\mathsf{EK}, \mathcal{L}), CSK_i)$, and output (m, sk_{ctr}).
- **Prove.consumer**$(PP, sk_{ctr}, \mathbf{req}) \to \mathbf{proof}$.
  - $\mathsf{SAS.Sign}(PP_{\mathrm{SAS}}, sk_{ctr}, \mathbf{req})$: set $sk_{ctr} = x_i$, calculate $h = \mathrm{H}_1(\mathbf{req} \parallel \omega)$, $A = \mathrm{H}_2(0 \parallel \omega)$, $B = \mathrm{H}_2(1 \parallel \omega)$, $\sigma = A^{x_i} B^{x_i h}$, where ω is the current period.
  - Output $\mathbf{proof} = (\mathbf{req}, \sigma, \omega)$.
- **ProofAgg.cloud**$(PP, \mathbf{PK}, \mathbf{Proof}) \to \mathbf{proof}_\Sigma$.
  - Collect $\mathbf{Proof} = (\mathbf{proof}_1, \ldots, \mathbf{proof}_l)$ under public keys $\mathbf{PK} = (pk_1, \ldots, pk_l)$. At the end of the period ω, check $\mathbf{proof}_i = (\mathbf{req}_i, \sigma_i, \omega_i)$ is valid. For $i \in [l]$:
    $\mathsf{SAS.Verify}(PP_{\mathrm{SAS}}, pk_i, \mathbf{req}_i, \sigma_i)$: set $pk_i = g_3^{x_i} = X_i$, compute $h_i = \mathrm{H}_1(\mathbf{req}_i \parallel \omega)$, $A = \mathrm{H}_2(0 \parallel \omega)$, $B = \mathrm{H}_2(1 \parallel \omega)$. If $\omega_i \neq \omega$ or $e(\sigma_i, g_3) \neq e(AB^{h_i}, X_i)$, halt.
  - $\mathsf{SAS.Aggregate}(PP_{\mathrm{SAS}}, \mathbf{PK}, \mathbf{Req}, \mathbf{Sig})$: set $\mathbf{Req} = (\mathbf{req}_1, \ldots, \mathbf{req}_l)$, $\mathbf{Sig} = (\sigma_1, \ldots, \sigma_l)$, and compute $\sigma_\Sigma = \prod_{i=1}^{l} \sigma_i$.
  - Output $\mathbf{proof}_\Sigma = (\mathbf{Req}, \sigma_\Sigma, \omega)$.
- **ProofVerify.owner**$(PP, \mathbf{PK}, \mathbf{proof}_\Sigma) \to 1$ or 0.
  - If ω is not the current period, or there is one public key that appears twice in **PK**, halt and output 0.
  - $\mathsf{SAS.AggVerify}(PP_{\mathrm{SAS}}, \mathbf{PK}, \mathbf{Req}, \sigma_\Sigma)$: for all $i \in [l]$, set $pk_i = g_3^{x_i} = X_i$, calculate $h_i = \mathrm{H}_1(\mathbf{req}_i \parallel \omega)$, $A = \mathrm{H}_2(0 \parallel \omega)$, $B = \mathrm{H}_2(1 \parallel \omega)$ and verify that

    $$e(\sigma_\Sigma, g_3) \stackrel{?}{=} e(A, \prod_{i=1}^{l} X_i) \cdot e(B, \prod_{i=1}^{l} X_i^{h_i}).$$

    If the equation holds, output 1. Otherwise, output 0.

6 Performance Evaluations

In this section, we give both theoretical and experimental analyses of the constructed FA-ABE.

6.1 Theoretical Analysis

Function Comparison. Table 1 compares four kinds of audits in FA-ABE with those in related work. (1) For the access control from the data owner to consumers, all schemes achieve data confidentiality that only authorized consumers have the decryption permission. (2) For the audit from consumers to the cloud, schemes in [6], [7], [9] outsource decryption operations to a semi-honest cloud provider, and [5], [11]–[14] further make the outsourced decryption result verifiable. Considering the cloud as a covert adversary, [4] provides the audit that the cloud cannot forge data, and FA-ABE achieves both data unforgeability and outsourced decryption verifiability simultaneously. (3) For the download control from the cloud to consumers, [15] realizes limited time access control, and [4] realizes the prevention of DDoS attacks on the DEM part of ciphertexts, but not the KEM part. FA-ABE achieves all three kinds of download control in the standard model. (4) For the resource consumption accountability from the owner to the cloud, [4] realizes the audit for the first time, and FA-ABE achieves more efficient aggregate accountability. In summary, our construction achieves all-sided accountability with high efficiency.

For consistency in the cost comparisons, we adopt the CP-ABE scheme [18] and the Pedersen commitment scheme [31] to instantiate related schemes. The BLS short signature [32] is used to instantiate the signature used in FOP [4][4].

Computation Cost Comparison. Table 3 shows the computation cost comparison of FA-ABE with related schemes.

4. Compared with standard signature algorithms, BLS signatures have very short length [33]. On an MNT curve (with $\alpha = 6$) over a 170-bit field, BLS signatures are 170 bits long, and provide security comparable to that of 1024-bit RSA [34] or 320-bit DSA [35]. Besides, BLS signatures can share the same elliptic curve in the experiment.

TABLE 3: Computation Cost Comparison†

Schemes	Encrypt.owner	Request&DecVerify.consumer	DecControl.cloud	ProofAgg.cloud	ProofVerify.owner
[9]	$(5l+2)$Exp + 1AES	1Exp + 1AES	$\|I\|$Exp + $(3\|I\|+1)$P	×	×
[11]	$(10l+6)$Exp + 1AES	4Exp + 1AES	$2\|I\|$Exp + $(6\|I\|+2)$P	×	×
[13], [14]	$(5l+4)$Exp + 1AES	3Exp + 1AES	$\|I\|$Exp + $(3\|I\|+1)$P	×	×
[15]	$(5l+2)$Exp + 1AES	3Exp + 1AES	$(\|I\|+1)$Exp + $(3\|I\|+3)$P	×	×
[4]	$(5l+4)$Exp + 2AES	$(\|I\|+1)$Exp + $(3\|I\|+3)$P + 3AES	1AES	×	$2n$P
Ours	$(5l+\kappa+4)$Exp + $(\kappa+1)$AES	4Exp + 2P + 1AES	$(\|I\|+1)$Exp + $(3\|I\|+3)$P	nExp + $2n$P	nExp + 3P

† Exp, P and AES denote a modular exponentiation, a pairing and an AES Encryption/Decryption computation, respectively. I, l, κ and n indicate the attribute set that satisfies the access policy, the number of rows of the matrix M for LSSS, the download limited time of the file in one period, the number of valid requests from data consumers, respectively.

TABLE 4: Communication Cost Comparison‡

Schemes	Original Ciphertext Created by the Owner	Data Transmitted out of the Cloud	Resource Consumption Proofs
[9], [15]	$(3l+1)\|\mathbb{G}\| + \|file\|$	$1\|\mathbb{G}_T\| + \|file\|$	×
[11]	$(6l+3)\|\mathbb{G}\| + \|file\|$	$1\|\mathbb{G}\| + 2\|\mathbb{G}_T\| + \|file\|$	×
[13], [14]	$(3l+2)\|\mathbb{G}\| + \|file\|$	$1\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + \|file\|$	×
[4]	$(3l+3)\|\mathbb{G}\| + 1\|\mathbb{Z}_p\| + \|file\|$	$(3l+3)\|\mathbb{G}\| + 1\|\mathbb{Z}_p\| + \|chal\| + \|file\|$	$n\|\mathbb{G}\|$
Ours	$\kappa\|\mathbb{Z}_p\| + (3l+\kappa+2)\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + \|file\|$	$1\|\mathbb{Z}_p\| + 1\|\mathbb{G}\| + 2\|\mathbb{G}_T\| + \|file\|$	$1\|\mathbb{G}\| + \|period\|$

‡ $|\mathbb{Z}_p|$, $|\mathbb{G}|$ and $|\mathbb{G}_T|$ denote the size of an element in $\mathbb{Z}_p$, $\mathbb{G}$ and $\mathbb{G}_T$, respectively. $|file|$, $|chal|$ and $|period|$ denotes the length of file, challenge value and time period representation, respectively. l, κ and n indicate the number of rows of the matrix M for LSSS, the download limited time of the file in one period, the number of valid requests from data consumers, respectively.

"Request&DecVerify.consumer" means the total computation of **Request.consumer**() and **DecVerify.consumer**(). (1) During the encrypt and upload phase, the computational cost of ABE encryption in our scheme is the same as [4], [13], [14], and less than [11]. In order to achieve limited time access control, FA-ABE requires additional κ modular exponentiations and AES encryptions. Nevertheless, the overhead is controllable. The owner may reduce κ by setting a shorter period, e.g., 1/hour instead of 24/day. (2) During the download and decrypt phase, our verification mechanism for outsourced decryption requires nearly half of the exponentiations used in [11], and one more modular exponentiation, two more pairings than [13], [14], to achieve data unforgeability simultaneously. Besides, the decryption efficiency is significantly better than FOP [4], in which the consumer needs to decrypt an original ABE ciphertext as well as an AES ciphertext to recover the challenge value. (3) During the prove and verify phase, the computation cost of proof verification for the owner in FA-ABE is reduced to only 3 pairings and a half of exponentiations in FOP [4]. In summary, our construction introduces slight cost to achieve all-sided accountability, and has a considerable advantage in efficiency compared with FOP [4].

Communication Cost Comparison. Table 4 compares the communication cost of FA-ABE with that in related schemes. We focus on three kinds of bandwidth consumption in the system: (1) The size of original ciphertext created by the owner. Compared with the optimal verifiable outsourced decryption technique [13], [14], we just add an element of $\mathbb{G}_T$ to achieve the data unforgeability and outsourced decryption verifiability, simultaneously. To realize limited time access control, κ elements of $\mathbb{Z}_p$ and $\mathbb{G}$ are introduced into the ciphertext, of which the overhead is also controllable. (2) The size of data transferred out of the cloud. In FOP [4], the cloud sends the consumer an ABE ciphertext (whose size is linear with the complexity of the access policy) and a challenge value. In FA-ABE, by contrast, the size of transferred data is constant, and almost equals to that of [9], [11], [13]–[15]. (3) The size of resource consumption proofs. Compared with FOP [4], all proofs in the same period are aggregated into one proof with constant size in FA-ABE. In summary, the size of original ciphertext is slightly expanded to achieve all-sided accountability, and the communication cost of the cloud responding data and resource consumption proofs is greatly decreased.

6.2 Experimental Analysis

We utilize the self-developed library libabe [7], [36] to evaluate the practical performance of FA-ABE, and make a comparison with FOP [4], which simultaneously achieves three kinds of audits in the pay-as-you-go model. We use the 224-bit MNT elliptic curve from Pairing-Based Cryptography library [37], the 128-bit AES-CBC, KDF and SHA-1 implementations from OpenSSL [38]. All running times are measured on a desktop with an Intel Core i7-3770 CPU @3.4 GHz and 8GB RAM running Ubuntu 16.04 LTS 64-bit.

Overall, we adopt similar experimental methods as those described in [9], [11]. We use access policies of type $(A_1\ and\ A_2\ and \cdots and\ A_l)$ to simulate the worst case, where A_i is an attribute. We set 20 distinct access policies with l increasing from 5 to 100, repeat each instance 20 times and eventually take the average. All instances are completely independent to each other. The test data to be encrypted is a PDF file with 1MB in size. For the resource consumption, supposing all consumers in the system generate 100 proofs every hour on average, we simulate the performance in a day (2400 proofs in total).

As depicted in Fig. 6, we show the time cost of the execution of each algorithm and the data size transferred out of the cloud. Fig. 6(a) illustrates the encryption time for the data owner of two constructions are almost the

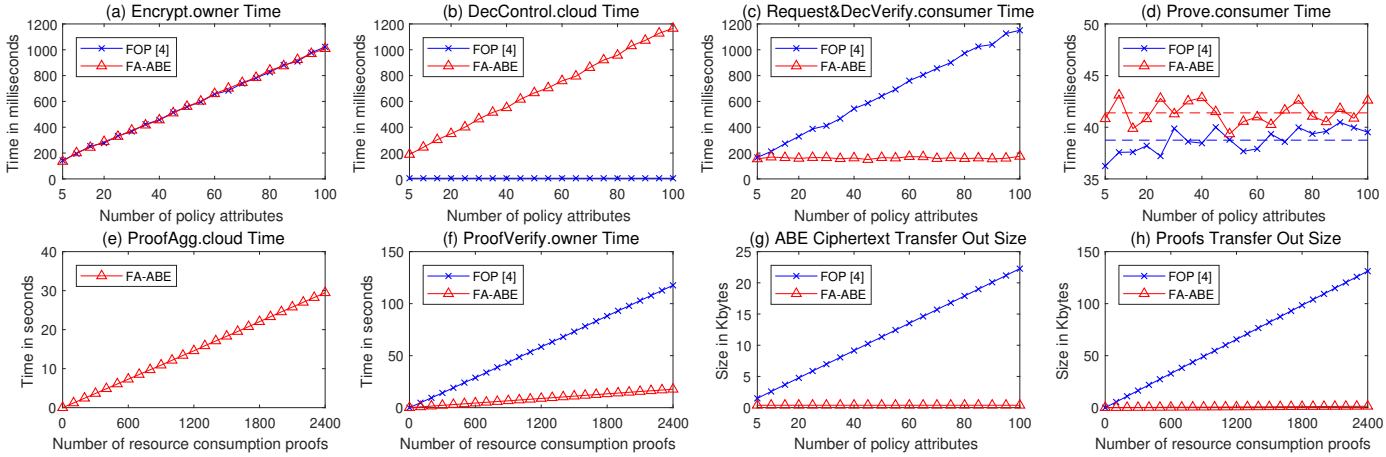


Fig. 6: Experimental Result. (a) Encrypt.owner Time. (b) DecControl.cloud Time. (c) Request&DecVerify.consumer Time. (d) Prove.consumer Time. (e) ProofAgg.cloud Time. (f) ProofVerify.owner Time. (g) ABE Ciphertext Transfer Out Size. (h) Proofs Transfer Out Size.

same. Fig. 6(c) shows that in FA-ABE, the computational cost of request, decryption and verification is remarkable lightweight, which only requires less than 200ms under any access policy. Fig. 6(d) and 6(f) illustrate that the proof generation time of FA-ABE is slightly longer than that of FOP on average, but FA-ABE saves much time for the data owner to verify consumption proofs. Large amount of complex computation (outsourced decryption and signature aggregation) is executed on the cloud as shown in Fig. 6(b) and 6(e). Fig. 6(g) and 6(h) indicate that the ABE ciphertext transfer overhead for a valid request and the proofs transfer overhead are reduced significantly in FA-ABE.

7 Conclusion

In this work, we propose fully accountable ABE for data sharing in the pay-as-you-go model. The construction not only achieves data confidentiality, non-interactive fine-grained access control and data unforgeability simultaneously, but also prevents DDoS attacks on both KEM part and DEM part of ciphertexts and prevents excessive data download. Complex operations are outsourced to the cloud securely, and the efficiency for resource consumption audit is optimized. We present a generic construction and prove the security in the standard model. The experimental results demonstrate that our construction is efficient and practical.

Acknowledgments

The authors wish to thank the anonymous reviewers for their valuable comments. This work was supported in part by the National Natural Science Foundation of China (Nos. 61632020, 61802392, 61772520, and 61972094), in part by the National Key Research and Development Program of China (No. 2017YFB0802705), in part by the Key Research Project of Zhejiang Province (No. 2017C01062), in part by the Beijing Natural Science Foundation (No. 4192067), and in part by the Beijing Municipal Science and Technology Project (Nos. Z191100007119007 and Z191100007119002). The corresponding author is Hui Ma.

References

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

[2] L. Cheung and C. C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 456–465.

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

[4] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2062–2074, Aug. 2018.

[5] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 679–692, Nov. 2017.

[6] R. Zhang, H. Ma, and Y. Lu, "Fine-grained access control system based on fully outsourced attribute-based encryption," *Journal of Systems and Software*, vol. 125, pp. 344–353, 2017.

[7] H. Ma, R. Zhang, G. Yang, Z. Song, K. He, and Y. Xiao, "Efficient fine-grained data sharing mechanism for electronic medical record systems with mobile devices," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2018.2844814.

[8] V. Shoup, "A proposal for an ISO standard for public key encryption," *IACR Cryptology ePrint Archive*, vol. 2001, p. 112, 2001.

[9] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *USENIX Secur. Symp.*, 2011.

[10] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," *J. Cryptology*, vol. 23, no. 2, pp. 281–343, 2010.

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

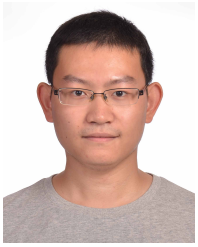
[12] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1384–1393, Jul. 2015.

[13] S. Lin, R. Zhang, H. Ma, and S. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 10, pp. 2119–2130, Oct. 2015.

[14] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 533–546, Sept. 2016.

[15] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 94–105, Jan. 2018.

[16] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.
[17] T. H. Yuen, J. K. Liu, M. H. Au, X. Huang, W. Susilo, and J. Zhou, "k-times attribute-based anonymous access control for cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2595–2608, Sept. 2015.
[18] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2013, pp. 463–474.
[19] Z. Liu, Z. Cao, and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 76–88, Jan. 2013.
[20] J. Ning, X. Dong, Z. Cao, and L. Wei, "Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud," in *Proc. ESORICS*, 2015, pp. 270–289.
[21] Y. Zhang, J. Li, D. Zheng, X. Chen, and H. Li, "Accountable large-universe attribute-based encryption supporting any monotone access structures," in *Proc. ACISP*, 2016, pp. 509–524.
[22] Y. Zhang, J. Li, D. Zheng, X. Chen, and H. Li, "Towards privacy protection and malicious behavior traceability in smart health," *Personal and Ubiquitous Computing*, vol. 21, no. 5, pp. 815–830, 2017.
[23] A. Beimel, "Secure schemes for secret sharing and key distribution," *Ph.D. dissertation, Faculty of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel*, 1996.
[24] K. Lee, D. H. Lee, and M. Yung, "Aggregating cl-signatures revisited: Extended functionality and better efficiency," in *Proc. FC*, 2013, pp. 171–188.
[25] S. Micali, M. O. Rabin, and S. P. Vadhan, "Verifiable random functions," in *IEEE Annu. Symp. Found. of Comp. Science*, 1999, pp. 120–130.
[26] N. Bitansky, "Verifiable random functions from non-interactive witness-indistinguishable proofs," in *Proc. TCC*, 2017, pp. 567–594.
[27] S. Jarecki and V. Shmatikov, "Handcuffing big brother: an abuse-resilient transaction escrow scheme," in *Proc. EUROCRYPT*, 2004, pp. 590–608.
[28] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Proc. PKC*, 2005, pp. 416–431.
[29] Y. Lindell and J. Katz, *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.
[30] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *Proc. CRYPTO*, 2010, pp. 631–648.
[31] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. CRYPTO*, 1991, pp. 129–140.
[32] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. ASIACRYPT*, 2001, pp. 514–532.
[33] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "A survey of two signature aggregation techniques," *RSA Cryptobytes*, vol. 6, no. 2, pp. 1–10, 2003.
[34] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[35] NIST. (Jul. 2013). Digital Signature Standard (DSS), FIPS 186-4. [Online]. Available: *https://csrc.nist.gov/publications/fips*
[36] H. Ma, R. Zhang, G. Yang, Z. Song, S. Sun, and Y. Xiao, "Concessive online/offline attribute based encryption with cryptographic reverse firewalls - secure and efficient fine-grained access control on corrupted machines," in *Proc. ESORICS*, 2018, pp. 507–526.
[37] B. Lynn. (Jun. 2013). PBC Library. [Online]. Available: *https://crypto.stanford.edu/pbc*
[38] OpenSSL Project. [Online]. Available: *https://www.openssl.org*



Ti Wang received the B.E. degree in information security from the Xidian University, Xi'an, China, in 2016. He is currently pursuing the PhD degree in information security with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He is currently working on the security mechanisms in cloud computing.

Hui Ma received his B.E. degree in information security from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2012. He received his Ph.D. degree in information security from the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2017. Now he is with Institute of Information Engineering, Chinese Academy of Sciences as an assistant professor. He is working on applied cryptography and the security mechanisms in cloud computing.

Yongbin Zhou is currently a full professor of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. He is also a professor with School of Cyber Security, University of Chinese Academy of Sciences. His main research interests include theories and technologies of network and information security.

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

Zishuai Song received the B.E. degree in information security from the Xidian University, China, in 2017. He is currently pursuing the Ph.D. degree in information security with the Institute of Information Engineering, Chinese Academy of Sciences. He is currently involved in the security mechanisms in cloud computing.