This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2016.2629510, IEEE Transactions on Computers

Provably Secure Key-Aggregate Cryptosystems with Broadcast Aggregate Keys for Online Data Sharing on the Cloud

Sikhar Patranabis, Yash Shrivastava and Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
{sikhar.patranabis, yash.shrivastava, debdeep}@cse.iitkgp.ernet.in

Abstract—Online data sharing for increased productivity and efficiency is one of the primary requirements today for any organization. The advent of cloud computing has pushed the limits of sharing across geographical boundaries, and has enabled a multitude of users to contribute and collaborate on shared data. However, protecting online data is critical to the success of the cloud, which leads to the requirement of efficient and secure cryptographic schemes for the same. Data owners would ideally want to store their data/files online in an encrypted manner, and delegate decryption rights for some of these to users, while retaining the power to revoke access at any point of time. An efficient solution in this regard would be one that allows users to decrypt multiple classes of data using a single key of constant size that can be efficiently broadcast to multiple users. Chu et al. proposed a key aggregate cryptosystem (KAC) in 2014 to address this problem, albeit without formal proofs of security. In this paper, we propose CPA and CCA secure KAC constructions that are efficiently implementable using elliptic curves and are suitable for implementation on cloud based data sharing environments. We lay special focus on how the standalone KAC scheme can be efficiently combined with broadcast encryption to cater to m data users and m' data owners while reducing the reducing the secure channel requirement from $\mathcal{O}(mm')$ in the standalone case to $\mathcal{O}(m + m')$.

Index Terms—Cloud Computing, Data Sharing, Data Security, Key-Aggregate Cryptosystem, Provable Security, Scalability, Broadcast Encryption, Semantic Security, CCA Security

1 Introduction

The recent advent of cloud computing has pushed the limits of data sharing capabilities for numerous applications that transcend geographical boundaries and involve millions of users. Governments and corporations today treat data sharing as a vital tool for enhanced productivity. Cloud computing has revolutionized education, healthcare and social networking. Perhaps the most exciting use case for cloud computing is its ability to allow multiple users across the globe share and exchange data, while saving the pangs of manual data exchanges, and avoiding the creation of redundant or out-of-date documents. Social networking sites have used the cloud to create a more connected world where people can share a variety of data including text and multimedia. Collaborative tools commonly supported by cloud platforms and are extremely popular since they lead to improved productivity and synchronization of effort. The impact of cloud computing has also pervaded the sphere of healthcare, with smartphone applications that allow remote monitoring and even diagnosis of patients. In short, cloud computing is changing various aspects of our lives in unprecedented ways.

Despite all its advantages, the cloud is susceptible to privacy and security attacks, that are a major hindrance to its wholesome acceptance as the primary means of data sharing in todays world. According to a survey carried out by IDC Enterprise Panel in August 2008 [1], Cloud users regarded security as the top challenge with 75% of surveyed users worried about their critical business and IT systems being vulnerable to attack. While security threats from external agents are widespread, malicious service providers must also be taken into consideration. Since online data almost always resides in shared environments (for instance, multiple virtual machines running on the same physical device), ensuring security and privacy on the cloud is a non trivial task. When talking about security and privacy of data in the cloud, it is important to lay down the requirements that a data sharing service must provide in order to be considered secure. We list down here some of the most primary requirements that a user would want in a cloud-based data sharing service:

- *Data Confidentiality*: Unauthorized users (including the cloud service provider), should not be able to access the data at any given time. Data should remain confidential in transit, at rest and on backup media.
- *User revocation*: The data owner must be able to revoke any user's access rights to data the without affecting other authorized users in the group.
- *Scalability and Efficiency*: Perhaps the biggest challenge faced by data management on the cloud is maintaining scalability and efficiency in the face of immensely large user bases and dynamically changing data usage patterns.
- *Collusion between entities*: Any data sharing service

in the cloud must ensure that even when certain malicious entities collude, they should still not be able to access any of the data in an unauthorized fashion.

A traditional way of ensuring data privacy is to depend on the server to enforce access control mechanisms [2]. This methodology is prone to privilege escalation attacks in shared data environments such as the cloud, where data corresponding to multiple users could reside on the same server. Current technology for secure online data sharing comes in two major flavors - trusting a third party auditor [3], or using the user's own key to encrypt her data while preserving anonymity [4]. In either case, a user would want a reliable and efficient cryptographic scheme in place, with formal guarantees of security, high scalability and ease of use. The main challenge in designing such a cryptosystem lies in effective sharing of encrypted data. A data sharing scheme on the cloud is only successful if data owners can delegate the access rights to their data efficiently to multiple users, who can then access the data directly from the cloud servers. Figure 1 describes a realistic online data sharing set-up on the cloud. Assume that a data owner Alice is using an online data sharing service such as Microsoft OneDrive [5] to store certain classes of data (here *class* may refer to any data structure such as a file, folder or any collection of these). She wishes to add an additional layer of security for her data by storing them in an encrypted fashion. Now, she intends to share a specific subset $\mathcal{S}$ of these documents with a set $\hat{S}$ of data users. For this, she needs to provide each of these users with decryption rights to specific classes of the data that they are authorized to access. The challenge therefore is to design a secure and efficient online *partial* data sharing scheme that allows Alice to perform this task in an efficient and secure manner.

A näive (and extremely inefficient) solution is to have a different decryption key for each message class, and share them accordingly with the designated users via secured channels. This scheme is not practically deployable for two major reasons. Firstly, the number of secret keys would grow with the number of data classes. Secondly, any user revocation event would require Alice to entirely re-encrypt the corresponding subset of data, and distribute the new set of keys to the other existing valid users. This makes the scheme inefficient and difficult to scale. Since the decryption key in public key cryptosystems is usually sent via a secure channel, smaller key sizes are desirable. Moreover, resource constrained devices such as wireless sensor nodes and smart phones cannot afford large expensive storage for the decryption keys either. An ideal scenario, as described in Figure 1, is where Alice can construct a single constant size decryption key $K_{\mathcal{S}}$ that combines the decryption rights to each of the data classes in $\mathcal{S}$, and then use a public key framework to broadcast this key to the target set of users $\hat{\mathcal{S}}$ in the form of a low overhead broadcast aggregate key $K_{(\mathcal{S},\hat{\mathcal{S}})}$. This scheme is efficient, avoids the use of secret channels which are costly and difficult to realize in practice, and is scalable to any arbitrary number of data classes and data users. In this paper, we attempt to build precisely such a data sharing framework that is provably secure and at the same time, efficiently implementable.

1.1 The Key-Aggregate Encryption Scheme

The most efficient proposition pertaining to our problem statement, to the best of our knowledge, is made in [6]. The proposition is to allow Alice to combine the decryption power of multiple data classes into a single key of constant size. Thus, while each class of data is encrypted using a different public key, a single decryption key of constant size is sufficient to decrypt any subset of these classes. This system is popularly known as the key-aggregate cryptosystem (KAC), and derives its roots from the seminal work on broadcast encryption by Boneh *et.al*. [7]. KAC may essentially be considered as a dual notion of broadcast encryption [7]. In broadcast encryption, a single ciphertext is broadcast among multiple users, each of whom may decrypt the same using their own individual private keys. In KAC, a single aggregate key is distributed among multiple users and may be used to decrypt ciphertexts encrypted with respect to different classes. For broadcast encryption, the focus is on having shorter ciphertexts and low overhead individual decryption keys, while in KAC, the focus is in having short ciphertexts and low overhead *aggregate keys*. However, KAC as proposed in [6] suffers from two major drawbacks, each of which we address in this paper.

1) Firstly, no concrete proofs of cryptographic security for KAC are provided by the authors of [6]. We note here that there exist significant differences in the fundamental constructs for broadcast encryption and key aggregate encryption. Broadcast encryption essentially involves two classes of parties - the broadcaster who broadcasts the secret key, and the data users who decrypt the broadcast message. On the other hand, KAC involves three parties - the data owner who encrypts and puts the data in the online sharing environment, the data users who access the data by decrypting it, and the trusted third party that generates the aggregate key. Thus a security framework for the security of KAC must be defined in order to establish the specific adversarial models against which KAC is secure.
2) Secondly, the scheme proposed in [6] does not explicitly address the issue of aggregate key distribution among multiple users. In a practical data sharing environment with millions of users, it is neither practical nor efficient to depend on the existence of dedicated one-to-one secure channels for key distribution. A public key based solution for broadcasting the aggregate key among an arbitrarily large number of users is hence desirable.

1.2 Related Work

In this section we present a brief overview of public and private key cryptographic schemes in literature for secure online data sharing. While many of them focus on key aggregation in some form or the other, very few have the ability to provide constant size keys to decrypt an arbitrary number of encrypted entities.

1.2.1 Hierarchical Encryption

One of the most popular techniques for access control in online data storage is to use a pre-defined hierarchy of secret

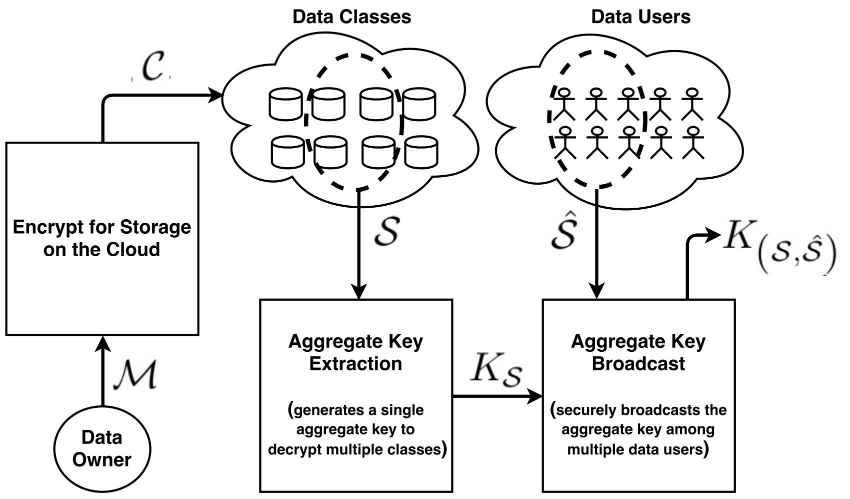
Data Classes
Data Users
$\mathcal{C}$
Encrypt for Storage on the Cloud
$\mathcal{S}$
$\hat{\mathcal{S}}$
$K_{(\mathcal{S},\hat{\mathcal{S}})}$
$\mathcal{M}$
Data Owner
Aggregate Key Extraction
(generates a single aggregate key to decrypt multiple classes)
$K_{\mathcal{S}}$
Aggregate Key Broadcast
(securely broadcasts the aggregate key among multiple data users)

Fig. 1: A Desirable Online Data Sharing Scheme

keys [8], [9], [10], [11] in the form of a tree-like structure, where access to the key corresponding to any node implicitly grants access to all the keys in the subtree rooted at that node. For instance, [12] uses repeated evaluations of a pseudo-random function/block cipher on a fixed secret to generate a tree hierarchy of symmetric keys. Some more advanced schemes [13], [14], [15] extend access control to cyclic and acyclic graphs. Several provably secure identity-based flavors of hierarchical encryption have also been studied extensively in [16], [17], [18], [19]. A major disadvantage of hierarchical encryption schemes is that granting access to only a selected set of branches within a given subtree warrants an increase in the number of granted secret keys. This in turn blows up the size of the key shared.

1.2.2 Compact Key Identity-Based Encryption

Identity-Based Encryption (IBE) is a public key-based encryption scheme in which the public key for any user is an identity-string corresponding to that user. Proposed initially in [20], IBE was concretized by the proposition of two very widely cited and popular IBEs - The Boneh-Franklin scheme [21] and Cocks' encryption scheme [22]. An IBE system comprises of a trusted private key generator that holds a master-secret key and issues a secret key to each user based on the user identity. Each user receives a message that has been encrypted using her id and some public parameters, and can decrypt the same using the secret key allotted to her by the trusted party. Compact key IBEs have been proposed in [23] and [24]. The former approach involves the use of random oracles while the latter shuns the use of oracles. Both these schemes allow aggregation of keys; however each key must come from a different identity division. Fuzzy IBE [25] allows for a single compact key to decrypt multiple ciphertexts, but they must have been encrypted under a closed set of identities, and the scheme does not work in practical scenarios for arbitrary identities.

1.2.3 Attribute Based Encryption

Attribute-based encryption (ABE) [26], [27], [28] allows each user to be identified by a set of attributes. An encrypted file stored in cloud can only be decrypted by an user who has access to the corresponding secret key. The secret key is securely transmitted to the user who satisfies the access control policies set by the data owner. A major drawback of this scheme is that each time the access right to a particular user is revoked the entire ciphertext has to be recrypted in the cloud. The idea of ABE has been extended to shared keys for user groups in [29] with the focus on collusion resistance and not on key size compression.

1.3 Our Contributions

The main contributions of this paper can be enumerated as follows:

1) In this paper we propose an efficiently implementable version of the basic key-aggregate cryptosystem (KAC) in [6] using asymmetric bilinear pairings. We prove our construction to be semantically secure against a non-adaptive adversary in the standard model under appropriate security assumptions. We also demonstrate that the construction is collusion resistant against any number of colluding parties.
2) We propose a CCA-secure fully collusion resistant construction for the basic KAC scheme with low overhead ciphertexts and aggregate keys. To the best of our knowledge, this is the first KAC construction in the cryptographic literature proven to be CCA secure in the standard model.
3) We demonstrate how the basic KAC framework may be efficiently extended and combined with broadcast encryption schemes for distributing the aggregate key among an arbitrary number of data users in a real-life data sharing environment. The extension has a secure channel requirement of $\mathcal{O}(m + m')$ for m data users and m' data owners, which is an improvement over the $\mathcal{O}(mm')$ requirement reported in [6]. In addition, the extended construction continues to have the same overhead for the public parameters, ciphertexts and aggregate keys, and does not require any secure storage for the aggregate keys, which are publicly broadcast.

4) Experimental results in an actual cloud environment are presented to validate the space and time complexity requirements, as well the network and communication requirements for our proposed constructions.

Table 1 compares various key delegation schemes discussed in Section 1.2 with our proposed KAC constructions for m' data owners and m data users. We point out here that both the original KAC [6] and our generalized KAC construction achieves the best overall performance and efficiency among all these schemes in terms of the decryption key size and the ciphertext overhead. However, our proposed generalized KAC construction requires much fewer secure channels due to its combination with broadcast encryption, that makes aggregate key distribution among multiple users more efficient and practically realizable.

2 Preliminaries

We begin by formally defining the framework key-aggregate cryptosystem (KAC). For clarity of presentation, we describe the framework in two parts. The basic framework focuses on generating the aggregate key for any arbitrary subset of data classes, while the extended framework aims to broadcast this aggregate key among arbitrarily large subsets of data users. We also outline the game based framework for formally proving the static security of these schemes. Finally, we state the complexity assumptions used for proving the security of these schemes.

2.1 Key-Aggregate Cryptosystem (KAC) : The Basic Framework

The basic KAC framework presented here is the same as that in [6] and is presented for completeness. KAC is an ensemble of five randomized polynomial-time algorithms. The system administrator is responsible for setting up the public parameters via the **SetUp** operation. A data owner willing to share her data using this system registers to receive her own public and private key pairs, generated using the **KeyGen** operation. The data owner is responsible for classifying each of her data files/messages into a specific class i. Each message is accordingly encrypted by an **Encrypt** operation and stored online in the cloud. When delegating the decryption rights to a specific subset of message classes, the data owner uses the **Extract** operation to generate a constant-size *aggregate decryption key* unique to that subset. Finally, an authorized data user can use this aggregate key to decrypt any message belonging to any class $i \in \mathcal{S}$. We now describe each of the five algorithms involved in KAC:

1) **SetUp**$(1^\lambda, m)$: Takes as input the number of data classes n and the security parameter λ. Outputs the public parameter $param$.
2) **KeyGen**(): Outputs the public key PK and the master-secret key msk for a data owner registering in the system.
3) **Encrypt**$(param, PK, i, \mathcal{M})$: Takes as input the public key parameter PK, the data class i and the plaintext data $\mathcal{M}$. Outputs the corresponding ciphertext $\mathcal{C}$.
4) **Extract**$(param, msk, \mathcal{S})$: Takes as input the master secret key and a subset of data classes $\mathcal{S} \subseteq \{1, 2, \cdots, n\}$. Computes the aggregate key $K_\mathcal{S}$ for all encrypted messages belonging to these subset of classes.
5) **Decrypt**$(param, \mathcal{C}, i, \mathcal{S}, K_\mathcal{S})$: Takes as input the ciphertext $\mathcal{C}$, the data class i and the aggregate key $K_\mathcal{S}$ corresponding to the subset $\mathcal{S}$ such that $i \in \mathcal{S}$. Outputs the decrypted message.

2.2 Security of Basic KAC : A Game Based Framework

In this paper, we propose a formal framework for proving the security of the basic KAC introduced in Section 2.1. We introduce a game between an attack algorithm $\mathcal{A}$ and a challenger $\mathcal{B}$, both of whom are given n, the total number of message classes, as input. The game proceeds through the following stages:

1) **Init**: Algorithm $\mathcal{A}$ begins by outputting a set $\mathcal{S}^* \subseteq \{1, 2, \cdots, n\}$ of data classes that it wishes to attack. Challenger $\mathcal{B}$ randomly chooses a message class $i \in \mathcal{S}^*$.
2) **SetUp**: Challenger $\mathcal{B}$ sets up the KAC system by generating the public parameter $param$, the public key PK and the master secret key msk. Since collusion attacks are allowed in our framework, $\mathcal{B}$ furnishes $\mathcal{A}$ with the aggregate key $K_{\overline{\mathcal{S}}^*}$ that allows $\mathcal{A}$ to decrypt any message class $j \notin \mathcal{S}^*$.
3) **Query Phase 1**: Algorithm $\mathcal{A}$ adaptively issues decryption queries $q_1, \cdots, q_v$ where a decryption query comprises of the tuple $(j, \mathcal{C})$, where $j \in \mathcal{S}^*$. The challenger responds with a valid decryption of $\mathcal{C}$.
4) **Challenge**: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages belonging to class i and provides them to $\mathcal{B}$. To generate the challenge, $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge to $\mathcal{A}$ as $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$, where $\mathcal{C}^*$ = **Encrypt**$(param, PK, i, \mathcal{M}_b)$.
5) **Query Phase 2**: Algorithm $\mathcal{A}$ continues to adaptively issue decryption queries $q_{v+1}, \cdots, q_{Q_D}$ where a decryption query now comprises of the tuple $(j, \mathcal{C})$ under the restriction that $\mathcal{C} \neq \mathcal{C}^*$. The challenger responds as in phase 1.
6) **Guess**: The adversary $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{A}$ wins the game.

The game above models an attack in the real world setting where users who do not have authorized access to the subset $\mathcal{S}^*$ collude (by compromising the knowledge of the aggregate key for different subsets) to try and expose a message in this subset. Note that the adversary $\mathcal{A}$ is non-adaptive; it chooses $\mathcal{S}$, and obtains the aggregate decryption key for all message classes outside of $\mathcal{S}$, before it even sees the public parameters $param$ or the public key PK. Let $Adv_{\mathcal{A},n}$ denote the probability that $\mathcal{A}$ wins the game when the challenger is given n as input. We next define the security of KAC against chosen ciphertext attacks (CCA) and chosen plaintext attacks (CPA) as follows:

- **CCA Security:** We say that a key-aggregate encryption system is (τ, ϵ, n, q_D) CCA secure if for all non-adaptive τ-time algorithms $\mathcal{A}$ that can make a total of

TABLE 1: A Comparative Summary of various Data Sharing Schemes for m' Data Owners and m Data Users

Scheme	Decryption Key Size	Ciphertext Size	Encryption Type	Secure Channels
Hierarchical Encryption [15]	Generally non-constant	Constant	Symmetric/Public key	$\mathcal{O}(mm')$
Compact Key Symmetric Encryption [30]	Constant	Constant	Symmetric	$\mathcal{O}(mm')$
Compact Key IBE [24]	Constant	Non-constant	Public key	$\mathcal{O}(m)$
Attribute-Based Encryption [26]	Non-constant	Constant	Public key	$\mathcal{O}(m)$
Basic KAC [6]	Constant	Constant	Public key	$\mathcal{O}(mm')$
Generalized KAC (Our Proposal)	Constant	Constant	Public key	$\mathcal{O}(m+m')$

q_D decryption queries, we have that $|Adv_{\mathcal{A},n} - \frac{1}{2}| < \epsilon$.

- **CPA Security:** We say that a key-aggregate encryption system is (τ, ϵ, n) CPA secure if it is $(\tau, \epsilon, n, 0)$ CCA secure.

2.3 Bilinear Pairings

In this paper, we make several references to bilinear non-degenerate mappings on elliptic curve sub-groups, popularly known in literature as *pairings*. Hence we begin by providing a brief background on bilinear pairing based schemes on elliptic curve subgroups. A pairing is a bilinear map defined over elliptic curve subgroups. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two (additive) cyclic elliptic curve subgroups of the same prime order q. Let $\mathbb{G}_T$ be a multiplicative group, also of order q with identity element 1. Also, let P and Q be points on the elliptic curve that generate the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. A mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is said to be a bilinear map if it satisfies the following properties:

- Bilinear: For all $P_1 \in \mathbb{G}_1, Q_1 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_q$, we have $e(aP_1, bQ_1) = e(P_1, Q_1)^{ab}$.
- Non-degeneracy: If P and Q be the generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively where neither group only contains the point at infinity, then $e(P, Q) \neq 1$.
- Computability: There exists an efficient algorithm to compute $e(P_1, Q_1)$ for all $P_1 \in \mathbb{G}_1$ and $Q_1 \in \mathbb{G}_2$.

2.4 Notations Used

This section introduces some notations that are used throughout this paper. We assume the existence of equi-prime order (q) elliptic curve subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$, along with their generators P and Q. We also assume the existence of a multiplicative cyclic group $\mathbb{G}_T$, also of order q with identity element 1. Let α be a randomly chosen element in $\mathbb{Z}_q$. For any point R in either $\mathbb{G}_1$ or $\mathbb{G}_2$, let $R_x = \alpha^x R$, where x is an integer. We denote by $Y_{R,\alpha,l}$ the set of $2l-1$ points $(R_1, R_2, \cdots, R_l, R_{l+2}, \cdots, R_{2l})$. Note that the term R_{l+1} is missing. We assume the existence of an efficiently computable asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Finally, any group element in $\mathbb{G}_1, \mathbb{G}_2$ or $\mathbb{G}_T$ is assumed to have size $\mathcal{O}(\eta_1)$, $\mathcal{O}(\eta_2)$ and $\mathcal{O}(\eta_T)$ respectively.

2.5 Complexity Assumption for Security

In this section, we introduce some complexity assumptions used to prove the security of our KAC constructions in this paper. We propose two complexity assumptions, both of which are extended versions of the generalized bilinear Diffie Hellman exponent (BDHE) assumption introduced in [7]:

- *The asymmetric decision l-BDHE problem*: Given an input $(H, I = (P, Q, Y_{P,\alpha,l}, Y_{Q,\alpha,l}), Z)$, where $H \in \mathbb{G}_2$ and $Z \in \mathbb{G}_T$, and the bilinear pairing e, decide if $Z = e(P_{l+1}, H)$.
- *The extended asymmetric decision l-BDHE problem*: Given an input $((H_1, H_2), I = (P, Q, Y_{P,\alpha,l}, Y_{Q,\alpha,l})(Z_1, Z_2))$, and the bilinear pairing e, where $H_1, H_2 \in \mathbb{G}_2$ and $Z_1, Z_2 \in \mathbb{G}_T$ decide if $(Z_1, Z_2) = (e(P_{l+1}, H_1), e(P_{l+1}, H_2))$.

Let $\mathcal{A}$ be a τ-time algorithm that takes an input challenge for asymmetric l-BDHE and outputs a decision bit $b \in \{0, 1\}$. We say that $\mathcal{A}$ has advantage ϵ in solving the asymmetric decision l-BDHE problem if

$$|Pr[\mathcal{A}(H, I, e(P_{l+1}, H)) = 0] - Pr[\mathcal{A}(I, Z) = 0]| \geq \epsilon$$

where the probability is over random choice of $H \in \mathbb{G}_2$, random choice of $Z \in \mathbb{G}_T$, random choice of $\alpha \in \mathbb{Z}_q$, and random bits used by $\mathcal{A}$. We refer to the distribution on the left as $\mathcal{L}_{\text{BDHE}}$ and the distribution on the right as $\mathcal{R}_{\text{BDHE}}$.

Again, let $\mathcal{B}$ be a τ-time algorithm that takes an input challenge for the extended asymmetric l-BDHE and outputs a decision bit $b \in \{0, 1\}$. We say that $\mathcal{B}$ has advantage ϵ in solving the extended asymmetric decision l-BDHE problem if

$$\begin{aligned}|Pr[\mathcal{B}\left((H_1, H_2), I, (e(P_{l+1}, H_1), e(P_{l+1}, H_2))\right) = 0]\\ -Pr[\mathcal{B}((H_1, H_2), I, (Z_1, Z_2)) = 0]| \geq \epsilon\end{aligned}$$

where the probability is over random choice of $H_1, H_2 \in \mathbb{G}_2$, random choice of $Z_1, Z_2 \in \mathbb{G}_T$, random choice of $\alpha_1, \alpha_2 \in \mathbb{Z}_q$, and random bits used by $\mathcal{B}$. We now refer to the distribution on the left as $\mathcal{L'}_{\text{BDHE}}$ and the distribution on the right as $\mathcal{R'}_{\text{BDHE}}$. We next state the following definitions.

Definition 1. *The* asymmetric decision (τ, ϵ, l)-BDHE assumption *holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if no* τ*-time algorithm has advantage at least* ϵ *in solving the asymmetric decision l-BDHE problem in* $(\mathbb{G}_1, \mathbb{G}_2)$.

Definition 2. *The* extended asymmetric decision (τ, ϵ, l)-BDHE assumption *holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if no* τ*-time algorithm has advantage at least* ϵ *in solving the extended asymmetric decision l-BDHE problem in* $(\mathbb{G}_1, \mathbb{G}_2)$.

Finally, it is quite evident that the extended asymmetric decision (τ, ϵ, l)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if the asymmetric decision (τ, ϵ, l)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$.

2.6 Signature Schemes

We briefly recall the standard definition of a signature scheme in the cryptographic literature. A signature scheme consists of the following polynomial-time algorithms:

1) $SigKeyGen(1^\lambda)$: Takes as input the security parameter λ and outputs a key pair (K_{SIG}, V_{SIG}), where K_{SIG} and V_{SIG} are the private signing key and public verification key respectively.
2) $Sign(K_{SIG}, M)$: Takes as input the signing key K_{SIG} and a message M. Outputs the corresponding signature-message pair (Sig, M).
3) $Verify(V_{SIG}, (Sig, M))$: Takes as input the verification key V_{SIG} and a signature-message pair (Sig, M). Outputs 1 if Sig is a valid signature for M under the signing key K_{SIG} and 0 otherwise.

We also recall that a signature scheme $(SigKeyGen, Sign, Verify)$ is said to be (τ, ϵ, q_S) strongly existentially unforgeable if no τ-time adversary, making at most q_S signature signature queries, fails to produce some new message-signature pair with probability at least $1 - \epsilon$ for ϵ negligible in the security parameter λ. For a more complete description, refer [31].

3 A Provably Secure Basic KAC Using Asymmetric Bilinear Pairings

In this section, we present the design of the basic key-aggregate cryptosystem introduced in [6] using asymmetric bilinear pairings that are practical and efficiently implementable, and formally prove its cryptographic security. The basic KAC construction serves to illustrate how a single data owner with n different classes of encrypted data online, can generate a single decryption key corresponding to any arbitrary subset $\mathcal{S} \subseteq \{1, \cdots, n\}$ of these data classes. We prove our construction to be non-adaptively CPA secure and fully collusion resistant against any number of colluding parties, under the asymmetric n-BDHE exponent assumption.

3.1 Construction

This section presents the basic KAC construction for a single data owner using asymmetric bilinear pairings. As mentioned in Section 2, we assume the existence of equi-prime order (for a λ-bit prime q) elliptic curve subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$, along with their generators P and Q. We also assume the existence of a multiplicative cyclic group $\mathbb{G}_T$, also of order q with identity element 1. Finally, we assume there exists an asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. The notations used in the forthcoming discussion are already introduced in Section 2.

SetUp$(1^\lambda, n)$: Randomly pick $\alpha \in \mathbb{Z}_q$. Output the system parameter as $param = (P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n})$. Discard α.

KeyGen(): Randomly pick $\gamma \in \mathbb{Z}_q$. Set the master secret key msk to γ. Let $PK_1 = \gamma P$ and $PK_2 = \gamma Q$. Set the public key $PK = (PK_1, PK_2)$. Output (msk, PK).

Encrypt$(param, PK, i, \mathcal{M})$: For a message $\mathcal{M} \in \mathbb{G}_T$ belonging to class $i \in \{1, 2, \cdots, n\}$, randomly choose $t \in \mathbb{Z}_q$. Output the ciphertext $\mathcal{C}$ as

$$\mathcal{C} = (c_0, c_1, c_2) = (tQ, t(PK_2 + Q_i), \mathcal{M} \cdot e(P_n, tQ_1))$$

Extract$(param, msk, \mathcal{S})$: For the subset of class indices $\mathcal{S}$, the aggregate key is computed as

$$K_{\mathcal{S}} = msk \sum_{j \in \mathcal{S}} P_{n+1-j} = \gamma \sum_{j \in \mathcal{S}} P_{n+1-j}$$

Note that this is indirectly equivalent to setting $K_{\mathcal{S}}$ to $\sum_{j \in \mathcal{S}} \alpha^{n+1-j} PK_1$.

Decrypt$(param, \mathcal{C}, i, K_{\mathcal{S}})$: Let $\mathcal{C} = (c_0, c_1, c_2)$. If $i \notin \mathcal{S}$, output $\perp$. Otherwise, set

$$\begin{aligned} a_{\mathcal{S}} &= \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i} \\ b_{\mathcal{S}} &= \sum_{j \in \mathcal{S}} P_{n+1-j} \end{aligned}$$

and return the decrypted message $\hat{\mathcal{M}}$ as:

$$\hat{\mathcal{M}} = c_2 \cdot \frac{e(K_{\mathcal{S}} + a_{\mathcal{S}}, c_0)}{e(b_{\mathcal{S}}, c_1)}$$

Observe that the above KAC construction is independent of the manner in which a data owner chooses to organize her data classes. Any KAC construction can support hierarchical data structures, since a data owner can create an aggregate key corresponding to all the data classes rooted at any internal node, which is then broadcast to the target user group.

3.2 Correctness

The proof of correctness of the basic KAC scheme is presented next.

$$\begin{aligned} \hat{\mathcal{M}} &= c_2 . \frac{e(K_{\mathcal{S}} + \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i}, c_0)}{e(\sum_{j \in \mathcal{S}} P_{n+1-j}, c_1)} \\ &= c_2 . \frac{e(\sum_{j \in \mathcal{S}} \gamma P_{n+1-j} + \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i}, tQ)}{e(\sum_{j \in \mathcal{S}} P_{n+1-j}, t(PK_2 + Q_i))} \\ &= c_2 . \frac{e(\sum_{j \in \mathcal{S}} \gamma P_{n+1-j}, tQ) e(\sum_{j \in \mathcal{S}} (P_{n+1-j+i}) - P_{n+1}, tQ)}{e(\sum_{j \in \mathcal{S}} P_{n+1-j}, tPK_2) e(\sum_{j \in \mathcal{S}} P_{n+1-j}, tQ_i))} \\ &= c_2 . \frac{e(\sum_{j \in \mathcal{S}} P_{n+1-j+i}, tQ)}{e(P_{n+1}, tQ) e(\sum_{j \in \mathcal{S}} P_{n+1-j}, tQ_i))} \\ &= c_2 . \frac{e(\sum_{j \in \mathcal{S}} P_{n+1-j+i}, tQ)}{e(P_{n+1}, tQ) e(\sum_{j \in \mathcal{S}} P_{n+1-j+i}, tQ))} \\ &= \mathcal{M} . \frac{e(P_n, tQ_1)}{e(P_{n+1}, tQ)} \\ &= \mathcal{M} \end{aligned}$$

3.3 Semantic Security

We now formally prove the CPA security of the basic KAC. We begin by stating the following theorem.

Theorem 1. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be bilinear elliptic curve subgroups of prime order q. For any positive integer n, the basic KAC handling n data classes is (τ, ϵ, n) CPA secure if the asymmetric decision (τ, ϵ, n)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$.*

Proof: Let $\mathcal{A}$ be a τ-time adversary such that $|Adv_{\mathcal{A},n} - \frac{1}{2}| > \epsilon$ for a KAC system parameterized with a given n. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the asymmetric n-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$. Algorithm $\mathcal{B}$ takes as input a random asymmetric n-BDHE challenge (H, I, Z) (where $I = (P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n})$ and Z is either

$e(P_{n+1}, H)$ or a random value in $\mathbb{G}_T$), and proceeds as follows.

Init: $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}$ of message classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a message class $i \in \mathcal{S}$ and lets $\mathcal{A}$ know of this choice.

SetUp: $\mathcal{B}$ should generate the public $param$ and the public key PK and provide them to $\mathcal{A}$. They are generated as follows.

- $\mathcal{B}$ sets $param$ is as $((P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n}))$.
- $\mathcal{B}$ randomly chooses $\gamma \in \mathbb{Z}_q$ and sets $PK = (PK_1, PK_2)$, where

$$\begin{aligned} PK_1 &= \gamma P - P_i \\ PK_2 &= \gamma Q - Q_i \end{aligned}$$

Note that this is equivalent to setting msk as $\gamma - \alpha^i$.

$\mathcal{B}$ also computes the collusion aggregate key $K_{\overline{\mathcal{S}}}$ as

$$K_{\overline{\mathcal{S}}} = \sum_{j \notin \mathcal{S}} (\gamma P_{n+1-j} - (P_{n+1-j+i}))$$

and provides it to $\mathcal{A}$. Note that this is indirectly equivalent to setting

$$\begin{aligned} K_{\overline{\mathcal{S}}} &= \sum_{j \notin \mathcal{S}} \alpha^{n+1-j}(\gamma P - P_i) \\ &= \sum_{j \notin \mathcal{S}} \alpha^{n+1-j} PK_1 \end{aligned}$$

as desired. Note that, $\mathcal{B}$ is aware that $i \notin \overline{\mathcal{S}}$ (implying $i \neq j$), and hence has all the resources to compute $K_{\overline{\mathcal{S}}}$.

Since P, Q, α and γ are all chosen uniformly at random, *the public parameters and the public key have an identical distribution to that in the actual construction.*

Challenge: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ in class i, and gives them to $\mathcal{B}$. In response, $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge to $\mathcal{A}$ as $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$, where $\mathcal{C}^* = (H, \gamma H, \mathcal{M}_b.Z)$. We claim that when $Z = e(P_{n+1}, H)$ (i.e. the input to $\mathcal{B}$ is a valid asymmetric n-BDHE tuple), then $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$ is a valid challenge to $\mathcal{A}$ as in a real attack. To see this, write $H = tQ$ for some unknown $t \in \mathbb{Z}_q$. Then we have

$$\begin{aligned} \gamma H &= t(\gamma Q - Q_i + Q_i) \\ &= t(PK_2 + Q_i) \\ \mathcal{M}_b.Z &= \mathcal{M}_b \cdot e(P_{n+1}, tQ) \\ &= \mathcal{M}_b e(P_n, tQ_1) \end{aligned}$$

Thus, by definition, $\mathcal{C}^*$ is a valid encryption of the message $\mathcal{M}_b$ in class i and hence, $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$ is a valid challenge to $\mathcal{A}$.

Guess: The adversary $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 0 (indicating that $Z = e(P_{n+1}, H)$). Otherwise, it outputs 1 (indicating that Z is a random element in $\mathbb{Z}_T$).

We now analyze the probability that $\mathcal{B}$ gives a correct output. If (I, Z) is sampled from $\mathcal{R}'_{\text{BDHE}}$, we have $Pr[\mathcal{B}(I, Z) = 0] = \frac{1}{2}$, while if (I, Z) is sampled from $\mathcal{L}'_{\text{BDHE}}$, $|Pr[\mathcal{B}(I, Z) = 0] - \frac{1}{2}| = |Adv_{\mathcal{A},n'} - \frac{1}{2}| \geq \epsilon$. This implies that $\mathcal{B}$ has advantage at least ϵ in solving the asymmetric n-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$. This concludes the proof of Theorem 1. Note that the proof is in the standard model does not require the use of random oracles. □

3.4 An Alternative Combination of the Message and the Secret

All our KAC constructions assume that any class of plaintext messages $\mathcal{M}$ may be efficiently embedded as elements in the multiplicative group G_T. This , however, may not be true for certain classes of data such as multimedia. A possible work around is to hash the secret value ρ used for encrypting $\mathcal{M}$ using a collision resistant hash function H, and then outputting $\mathcal{M} \odot H(\rho)$ in the ciphertext (here $\odot$ denotes an appropriate operator). Additionally, it is preferable to choose H from the family of *smooth projective hash functions* [32], that do not require the use of random oracles to prove security and can be efficiently designed to be collision resistant [33].

4 Chosen Ciphertext Secure Basic KAC

We now demonstrate how to modify the basic KAC proposed in Section 3.1 to obtain chosen ciphertext security. The resulting KAC system is proven to be CCA secure in the standard model without using random oracles. To the best of our knowledge, this is the first CCA secure KAC construction proposed in the cryptographic literature.

4.1 Additional Requirements for CCA Security

We have the following additional requirements for the CCA secure basic KAC:

- A signature scheme $(SigKeyGen, Sign, Verify)$ as defined in Section 2.6.
- A collision resistant hash function for mapping verification keys to $\mathbb{Z}_q$.

For simplicity of presentation, we assume here that the signature verification keys are encoded as elements of $\mathbb{Z}_q$. We avoid any further mention of the hash function in the forthcoming discussion, since it is implicitly assumed that any signature value we refer to is essentially the hash value corresponding to the original signature.

4.2 Construction

We now present the CCA secure construction for basic KAC. The security of our construction for n data classes is based on the asymmetric $(n+1)$-BDHE assumption, instead of the asymmetric n-BDHE assumption (as was for the semantically secure basic construction). For consistency of notation, we describe the construction for $n-1$ users instead of n users, such that the security assumption is still the asymmetric n-BDHE assumption as before.

SetUp$(1^\lambda, n-1)$: Randomly pick $\alpha \in \mathbb{Z}_q$. Output the system parameter as $param = (P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n}))$. Discard α.

KeyGen(): Randomly pick $\gamma \in \mathbb{Z}_q$. Set the master secret key msk to γ. Let $PK_1 = \gamma P$ and $PK_2 = \gamma Q$. Set the public key $PK = (PK_1, PK_2)$. Output (msk, PK).

Encrypt$(param, PK, i, \mathcal{M})$: Run the $SigKeyGen$ algorithm to obtain a signature signing key K_{SIG} and a verification key $V_{SIG} \in \mathbb{Z}_q$. Then, randomly choose $t \in \mathbb{Z}_q$ and set

$$c_0 = tQ \quad \text{and} \quad c_1 = t(PK_2 + Q_i + V_{SIG}Q_n)$$
$$c_2 = \mathcal{M} \cdot e(P_n, tQ_1)$$
$$\mathcal{C} = (\mathcal{C}' = (c_0, c_1, c_2), Sign(\mathcal{C}', K_{SIG}), V_{SIG})$$

Output the ciphertext $\mathcal{C}$.

Extract$(param, msk, \mathcal{S})$: For the subset of class indices $\mathcal{S}$, the aggregate key is computed as

$$K_{\mathcal{S}} = msk \sum_{j \in \mathcal{S}} P_{n+1-j} = \gamma \sum_{j \in \mathcal{S}} P_{n+1-j}$$

Decrypt$(param, \mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}})$: Let $\mathcal{C} = (\mathcal{C}', \sigma, V_{SIG})$. Verify that σ is a valid signature of $\mathcal{C}'$ under the key V_{SIG}. If not, output $\perp$. Also, if $i \notin \mathcal{S}$, output $\perp$. Otherwise, set

$$SIG_{\mathcal{S}} = \sum_{j \in \mathcal{S}} V_{SIG} P_{2n+1-j}$$
$$a_{\mathcal{S}} = \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i}$$
$$b_{\mathcal{S}} = \sum_{j \in \mathcal{S}} P_{n+1-j}$$

Note that these can be computed as $1 \leq i, j \leq n-1$. Next, pick a random $w \in \mathbb{Z}_q$ and set two entities $\hat{h}_1$ and $\hat{h}_2$ as

$$\hat{h}_1 = K_{\mathcal{S}} + SIG_{\mathcal{S}} + a_{\mathcal{S}} + w(PK_1 + P_i + V_{SIG}P_n)$$
$$\hat{h}_2 = b_{\mathcal{S}} + wP$$

Output the decrypted message

$$\hat{\mathcal{M}} = c_2 \cdot \frac{e(\hat{h}_1, c_0)}{e(\hat{h}_2, c_1)}$$

The proof of correctness of this scheme is very similar to the proof for the basic dynamic KAC scheme presented in Section 3.1. Note that the ciphertext size is still constant and everything else, including the public and private parameters, as well as the aggregate key, remains unchanged. The main change from the original scheme is in the fact that decryption requires a randomization value $w \in \mathbb{Z}_q$. This randomization makes sure that that the pair $(\hat{h}_1, \hat{h}_2)$ is chosen from the following distribution

$$(x(PK_1 + P_i + V_{SIG}P_n) - P_{n+1}, xP)$$

where x is chosen uniformly from $\mathbb{Z}_q$. This can be readily observed by setting $x = w + \sum_{j \in \mathcal{S}} \alpha^{n+1-j}$ (if w is uniformly random in $\mathbb{Z}_q$, so is x). *This randomization is a vital aspect from the point of view of CCA-security.* Note that the distribution $(\hat{h}_1, \hat{h}_2)$ depends only on the message class i for the message m to be decrypted and is independent of the subset $\mathcal{S}$ to which i belongs. However, decryption using only this distribution (that is without $K_{\mathcal{S}}$) requires the knowledge of P_{n+1}, which is not available and is hard to compute from $param$ under the BDHE assumption in $\mathbb{G}_1$.

Observe that, as in the basic construction, the public parameter for the CCA secure construction also comprises of $\mathcal{O}(n)$ group elements, and the aggregate key $K_{\mathcal{S}}$ as well as the public key PK consist of $\mathcal{O}(1)$ group elements each. Despite the presence of the additional components from the signature scheme, the ciphertext size is still $\mathcal{O}(1)$ group elements. Thus overall, the space complexities for various components of the scheme remain as in the basic construction.

4.3 Security

We state and prove the following theorem:

Theorem 2. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be bilinear elliptic curve subgroups of prime order q. For any positive integer n, the modified basic KAC handling $n-1$ data classes is $(\tau, \epsilon_1 + \epsilon_2, n-1, q_D)$ CCA-secure if the asymmetric decision (τ, ϵ_1, n)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ and the signature scheme is $(\tau, \epsilon_2, 1)$ strongly existentially unforgeable.*

Proof: Let $\mathcal{A}$ be a τ-time adversary such that $|Adv_{\mathcal{A},n-1} - \frac{1}{2}| > \epsilon_1 + \epsilon_2$. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ_1 in solving the asymmetric n-BDHE problem in $\mathbb{G}$. Algorithm $\mathcal{B}$ takes as input a random asymmetric n-BDHE challenge (H, I, Z) (where $I = (H, P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n})$, and Z is either $e(P_{n+1}, H)$ or a random value in $\mathbb{G}_T$), and proceeds as follows.

Init: $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}^*$ of message classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a message class $i \in \mathcal{S}^*$ and lets $\mathcal{A}$ know of this choice.

SetUp: $\mathcal{B}$ should generate the public $param$ and the public key PK, and provide them to $\mathcal{A}$. $\mathcal{B}$ first runs the $SigKeyGen$ algorithm to obtain a signature signing key K^*_{SIG} and a corresponding verification key $V^*_{SIG} \in \mathbb{Z}_q$. $\mathcal{B}$ generates the following:

- $param$ is set as $(P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n})$.
- Set $PK = (PK_1, PK_2)$, where PK_1 and PK_2 are computed as

$$PK_1 = \gamma P - V^*_{SIG}P_n - P_i$$
$$PK_2 = \gamma Q - V^*_{SIG}Q_n - Q_i$$

for γ chosen uniformly at random from $\mathbb{Z}_q$. Note that this is equivalent to setting $msk = \gamma - \alpha^i - \alpha^n V^*_{SIG}$.

$\mathcal{B}$ computes the collusion aggregate key $K_{\overline{\mathcal{S}^*}}$ as

$$K_{\overline{\mathcal{S}^*}} = \sum_{j \notin \mathcal{S}^*} (\gamma P_{n+1-j} - V^*_{SIG}P_{2n+1-j} - P_{n+1-j+i})$$

Note that $K_{\overline{\mathcal{S}^*}}$ is equal to $\sum_{j \notin \mathcal{S}^*} \alpha^{n+1-j} PK_1$, in accordance with the specification provided by the scheme. Moreover, $\mathcal{B}$ is aware that $i \notin \overline{\mathcal{S}^*}$ (implying $i \neq j$). Also, $j \neq n$ as $1 \leq j \leq n-1$. Hence, $\mathcal{B}$ has all the resources to compute $K_{\overline{\mathcal{S}^*}}$.

Since P, Q, α, and γ are chosen uniformly at random, *the public parameters and the public key have an identical distribution to that in the actual construction.*

Query Phase 1: $\mathcal{A}$ now issues decryption queries. Let $(j, \mathcal{C})$ be a decryption query where $j \in \mathcal{S}^*$. Let $\mathcal{C} = ((c_0, c_1, c_2), \sigma, V_{SIG})$. $\mathcal{B}$ first runs $Verify$ to check if the signature σ is valid on (c_0, c_1, c_2) using V_{SIG}. If invalid,

$\mathcal{B}$ returns $\perp$. If $V_{SIG} = V^*_{SIG}$, $\mathcal{B}$ outputs a random bit $b \in \{0,1\}$ and *aborts* the simulation. We show later that the probability of this event is negligible under the assumption that the signature is strongly existentially unforgeable. Otherwise, $\mathcal{B}$ picks a random $x \in \mathbb{Z}_q$ and sets

$$\begin{aligned}\hat{h}_0 &= (V_{SIG} - V^*_{SIG})P_n + P_j - P_i\\ \hat{h'}_0 &= (V_{SIG} - V^*_{SIG})^{-1}(P_{j+1} - P_{i+1})\\ \hat{h}_2 &= xP + (V_{SIG} - V^*_{SIG})^{-1}P_1\\ \hat{h}_1 &= \gamma\hat{h}_2 + x\hat{h}_0 + \hat{h'}_0\end{aligned}$$

$\mathcal{B}$ then responds with the decrypted message

$$\mathcal{M}' = c_2 \cdot \frac{e(\hat{h}_1, c_0)}{e(\hat{h}_2, c_1)}$$

It can be easily shown by choosing $x' = x + \alpha(V_{SIG} - V^*_{SIG})^{-1}$ that $\mathcal{B}$'s response is a valid decryption of the ciphertext $\mathcal{C}$ queried by $\mathcal{A}$, as in a real attack game.

Challenge: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages belonging to class i and provides them to $\mathcal{B}$. $\mathcal{B}$ in turn randomly picks $b \in \{0,1\}$, and sets

$$\begin{aligned}\mathcal{C}' &= (H, \gamma H, \mathcal{M}_b \cdot Z)\\ \mathcal{C}^* &= (\mathcal{C}, Sign(\mathcal{C}', K^*_{SIG}), V^*_{SIG})\end{aligned}$$

The challenge posed to $\mathcal{A}$ is $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$. It is easy to show that when $Z = e(P_{n+1}, H)$, $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$ is a valid challenge to $\mathcal{A}$ as in a real attack.

Query Phase 2: Same as in Query Phase 1.

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

Quite evidently, if (I, Z) is sampled from $\mathcal{R}'_{\text{BDHE}}$, $Pr[\mathcal{B}(I,Z) = 0] = \frac{1}{2}$. Let **abort** be the event that $\mathcal{B}$ aborted the simulation. Now when (I, Z) is sampled from $\mathcal{L}'_{\text{BDHE}}$, we have

$$|Pr[\mathcal{B}(I,Z) = 0] - \frac{1}{2}| > (\epsilon_1 + \epsilon_2) - Pr[\textbf{abort}]$$

This essentially implies that $\mathcal{B}$ has advantage at least $\epsilon_1 + \epsilon_2 - Pr[\textbf{abort}]$ in solving the asymmetric n-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

It is left to bound the probability that $\mathcal{B}$ aborts the simulation upon one of the decryption queries by $\mathcal{A}$. We claim that $Pr[\textbf{abort}] < \epsilon_2$; otherwise one can use $\mathcal{A}$ to forge signatures with probability at least ϵ_2. A very brief proof of this may be stated as follows. We may construct a simulator that knows the master secret key and receives K^*_{SIG} as a challenge in an existential forgery game. $\mathcal{A}$ can then cause an abort by producing a query that leads to an existential forgery under K^*_{SIG} on some ciphertext. Our simulator uses this forgery to win the existential forgery game. Only one chosen message query is made by the adversary during the game to generate the signature corresponding to the challenge ciphertext. Thus, $Pr[\text{abort}] < \epsilon_2$, implying $\mathcal{B}$ has advantage at least ϵ_1 in solving the asymmetric n-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$. This completes the proof of Theorem 2. □

5 Extended KAC with Aggregate Key Broadcast

The KAC constructions presented in Sections 3 and 4 require the aggregate keys to be transmitted to data users via secure channels. However, in a real world data sharing setup with m data users and m' data owners, such a solution requires the existence of $\mathcal{O}(mm')$ secure channels, which is extremely costly. In addition, the dynamically growing nature of the network implies that the requirement for secure channels increases in a multiplicative fashion with every new data owner/user joining the network. This makes the basic KAC scheme inconvenient for large scale deployment despite its cryptographically secure aggregate key generation properties.

In this section, we develop a novel mechanism for public-key based aggregate key distribution that reduces the secure channel requirement to $\mathcal{O}(m+m')$ from $\mathcal{O}(mm')$. We use *broadcast encryption*, which is a well known technique in public key cryptography, to efficiently distribute the aggregate keys among multiple users in a secure fashion. Our extended KAC construction combines the basic KAC instance presented in Section 3.1 with the public key based broadcast encryption system presented in [7] to build a fully public key based online data sharing scheme.

5.1 The Framework for Extended KAC

The framework for extended KAC with aggregate key broadcast is presented below:

1) **SetUp**$(1^\lambda, n, m)$: Takes as input the number of data classes n, the number of users m and the security parameter λ. Outputs the public parameter $param$.
2) **OwnerKeyGen**(): Outputs the public key PK, the master-secret key msk and the broadcast secret key bsk for a data owner registering in the system.
3) **OwnerEncrypt**$(param, PK, i, \mathcal{M})$: Takes as input a data class $i \in \{1, \cdots, n\}$ and the plaintext data $\mathcal{M}$. Outputs a partially encrypted ciphertext $\mathcal{C}'$. Note that $\mathcal{C}'$ is not the final ciphertext and is not exposed to the outside world. It is sent to the system administrator via a secure channel for further modification as described next. Note here that any instantiation of this scheme must ensure that the partial ciphertext $\mathcal{C}'$ is protected using suitable randomizations so as to leak nothing about the underlying plaintext data $\mathcal{M}$ during transmission to the system administrator.
4) **SystemEncrypt**$(\mathcal{C}', msk, bsk)$: Takes as input the partially encrypted ciphertext $\mathcal{C}'$, the master secret key msk and the broadcast secret key bsk. Outputs the final ciphertext $\mathcal{C}$ whih is made available on the cloud. This step is carried out by the system administrator, who is a trusted third party.
5) **UserKeyGen**$(param, msk, \hat{i})$: Takes as input the data user id $\hat{i} \in \{1, \cdots, m\}$ and outputs the corresponding secret key $d_{\hat{i}}$.
6) **Extract**$(param, msk, \mathcal{S})$: Takes as input the master secret key msk and a subset of data classes $\mathcal{S} \subseteq \{1, \cdots, n\}$. Computes the aggregate key $K_{\mathcal{S}}$ for all encrypted messages belonging to these subset of classes, and passes it as input to the **Broadcast**

algorithm for generating the broadcast aggregate key.

7) **Broadcast**$(param, K_{\mathcal{S}}, \hat{\mathcal{S}}, PK, bsk)$: Takes as input the aggregate key $K_{\mathcal{S}}$ and the target subset of users $\hat{\mathcal{S}} \subseteq \{1, \cdots, m\}$. Outputs a single *broadcast aggregate key* $K_{(\mathcal{S},\hat{\mathcal{S}})}$ that allows any user $\hat{i} \in \hat{\mathcal{S}}$ to decrypt all encrypted data/messages classified into any class $i \in \mathcal{S}$.
8) **Decrypt**$(param, \mathcal{C}, K_{(\mathcal{S},\hat{\mathcal{S}})}, i, \hat{i}, d_{\hat{i}}, \mathcal{S}, \hat{\mathcal{S}})$: The decryption algorithm now takes, besides the ciphertext $\mathcal{C}$ and the corresponding data class $i \in \mathcal{S}$, a valid user id $\hat{i} \in \hat{\mathcal{S}}$. It also takes as input the broadcast aggregate key $K_{(\mathcal{S},\hat{\mathcal{S}})}$ and the secret key $d_{\hat{i}}$. The algorithm outputs the decrypted message.

We note that the secure channel requirement is one per data owner (in **OwnerEncrypt**) and one per data user (in **UserKeyGen**). Thus the overall secure channel requirement is $\mathcal{O}(m + m')$. Observe that the main challenge to be tackled in realizing this scheme is combining the original aggregate key $K_{\mathcal{S}}$ with the broadcast secret to obtain the final broadcast aggregate key $K_{(\mathcal{S},\hat{\mathcal{S}})}$.

5.2 Security of Extended KAC: A Game Based Framework

We also define the formal framework for proving the security of the extended KAC proposed in Section 5.1 via the following game between an attack algorithm $\mathcal{A}$ and a challenger $\mathcal{B}$:

1) **Init**: Algorithm $\mathcal{A}$ begins by outputting a set $\mathcal{S}^* \subseteq \{1, 2, \cdots, n\}$ of data classes and a set $\hat{\mathcal{S}}^* \subseteq \{1, 2, \cdots, m\}$ of users that it wishes to attack. Challenger $\mathcal{B}$ randomly chooses a message class $i \in \mathcal{S}^*$.
2) **SetUp**: $\mathcal{B}$ sets up the KAC system by generating the public parameter $param$, the public key PK and the master secret key msk and the broadcast secret key bsk. Since collusion attacks are allowed in our framework, $\mathcal{B}$ furnishes $\mathcal{A}$ with all the private user keys $d_{\hat{j}}$ for $\hat{j} \notin \hat{\mathcal{S}}^*$. In addition, $\mathcal{A}$ also gets the aggregate key $K_{(\overline{\mathcal{S}^*},\hat{\mathcal{S}}^*)}$ that allows any user in $\hat{\mathcal{S}}^*$ to decrypt any message class $j \notin \mathcal{S}^*$.
3) **Query Phase 1**: $\mathcal{A}$ adaptively issues decryption queries $q_1, \cdots, q_v$ where a decryption query comprises of the tuple $(j, \mathcal{C})$, where $j \in \mathcal{S}^*$. The challenger responds with a valid decryption of $\mathcal{C}$.
4) **Challenge**: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages belonging to class i and provides them to $\mathcal{B}$. To generate the challenge, $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge to $\mathcal{A}$ as $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$, where $\mathcal{C}^*$ = **Encrypt**$(param, PK, i, \mathcal{M}_b)$.
5) **Query Phase 2**: $\mathcal{A}$ continues to adaptively issue decryption queries $q_{v+1}, \cdots, q_{Q_D}$ where a decryption query now comprises of the tuple $(j, \mathcal{C})$ under the restriction that $\mathcal{C} \neq \mathcal{C}^*$. $\mathcal{B}$ responds as in phase 1.
6) **Guess**: The adversary $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{A}$ wins the game.

The game above models an attack involving two different kinds of collusion. The first collusion is by all users not in $\hat{\mathcal{S}}^*$ who collude to try and expose an aggregate key that is broadcast for users in $\hat{\mathcal{S}}^*$ only. The second collusion is by users in $\hat{\mathcal{S}}^*$ who collude (by compromising the knowledge of the aggregate key for different subsets) to try and expose a message class in $\mathcal{S}^*$. The CPA and CCA security definitions of the extended scheme are similar to that for the basic scheme described in Section 2.2.

6 Construction for Extended KAC

In this section, we present a construction to realize the extended KAC scheme described in Section 5.1. The construction is inspired by techniques presented in [7]. Our construction uses a core building block that supports B data classes and B data users. The idea is to run $(A \times \hat{A})$ instances of this block in parallel, such that the overall system can handle $n = A \times B$ data classes and $m = \hat{A} \times B$ data users The building blocks share the same set of public parameters, but use their own set of private and public key components. The construction trades off the public parameter size with the public key size and the aggregate key size, while still maintaining constant ciphertext overhead. The construction is inspired by the two-tier broadcast encryption construction presented in [7].

6.1 The Construction

The extended KAC construction is presented below:

SetUp$_B(1^\lambda, n, m)$: Randomly pick $\alpha \in \mathbb{Z}_q$. Output the system parameter as

$$param = (P, Q, Y_{P,\alpha,B}, Y_{Q,\alpha,B})$$

Discard α. Compute $A = \lceil n/B \rceil$ and $\hat{A} = \lceil m/B \rceil$.

OwnerKeyGen(): Randomly pick two sets of values $\gamma_1^1, \cdots, \gamma_1^A \in \mathbb{Z}_q$, and $\gamma_2^1, \cdots, \gamma_2^{\hat{A}} \in \mathbb{Z}_q$. Set

$$msk_1 = \left(\gamma_1^1, \cdots, \gamma_1^A\right) \text{ and } msk_2 = \left(\gamma_2^1, \cdots, \gamma_2^{\hat{A}}\right)$$

Next, define for $1 \leq a \leq A$ and $1 \leq \hat{a} \leq \hat{A}$

$$\begin{array}{lcl} PK_1^a = \gamma_1^a P & , & PK_2^a = \gamma_1^a Q \\ PK_3^{\hat{a}} = \gamma_2^{\hat{a}} P & , & PK_4^{\hat{a}} = \gamma_2^{\hat{a}} Q \end{array}$$

and set

$$\begin{array}{rcl} PK_1 & = & \left(PK_1^1, \cdots, PK_1^A\right) \\ PK_2 & = & \left(PK_2^1, \cdots, PK_2^A\right) \\ PK_3 & = & \left(PK_3^1, \cdots, PK_3^{\hat{A}}\right) \\ PK_4 & = & \left(PK_4^1, \cdots, PK_4^{\hat{A}}\right) \end{array}$$

Output the master secret key as $msk = (msk_1, msk_2)$ and the public key as $PK = (PK_1, PK_2, PK_3, PK_4)$. Also output the secret broadcast key $bsk = \gamma_3$ chosen uniformly at random from $\mathbb{Z}_q$.

OwnerEncrypt$(param, PK, i, \mathcal{M})$: Compute $a = \lceil i/B \rceil$ and $b = i \bmod B+1$. Let PK_2^a be the ath component of PK_2.

Randomly choose $t \in \mathbb{Z}_q$ and output the partial ciphertext $\mathcal{C}'$ as

$$\begin{aligned}\mathcal{C}' &= (c_0, c_1', c_2, c_3)\\ &= (tQ, tPK_2^a, t(PK_2^a + Q_b), \mathcal{M} \cdot e(P_B, tQ_1))\end{aligned}$$

SystemEncrypt$(\mathcal{C}', i, msk, bsk)$: Takes as input the partially encrypted ciphertext $\mathcal{C}' = (c_0, c_1', c_2, c_3)$, the master secret key $msk = (msk_1, msk_2)$ and the broadcast secret key bsk. The additional information required here is the class id i. Let $a = \lceil i/B \rceil$ and msk_1^a be the a^{th} component of msk_1. Output the final ciphertext $\mathcal{C}$ as:

$$\begin{aligned}\mathcal{C} &= (c_0, c_1, c_2, c_3)\\ &= (c_0, c_1' - (bsk \cdot msk_1^a)Q, c_2, c_3)\\ &= (tQ, (t - bsk)PK_2^a, t(PK_2^a + Q_b), \mathcal{M} \cdot e(P_B, tQ_1))\end{aligned}$$

UserKeyGen$(param, msk, \hat{i})$: Compute $\hat{a} = \lceil \hat{i}/B \rceil$ and $\hat{b} = \hat{i} \bmod B + 1$. Let $msk_2^{\hat{a}}$ be the $\hat{a}^{\text{th}}$ component of msk_2. Output the private key for user with id $\hat{i}$ as:

$$d_{\hat{i}} = msk_2^{\hat{a}} P_{\hat{b}} = \gamma_2^{\hat{a}} P_{\hat{b}}$$

Note that this is indirectly equivalent to setting $d_{\hat{i}}$ to $\alpha^{\hat{b}} PK_3^{\hat{a}}$.

Extract$(param, msk, \mathcal{S})$: Let $msk_1 = (msk_1^1, \cdots, msk_1^A)$. For the subset of class indices $\mathcal{S}$ and $1 \le a \le A$, define

$$\mathcal{S}_a = \{i \bmod B + 1 | i \in \mathcal{S}, \lceil i/B \rceil = a\}$$

Next, for $1 \le a \le A$, compute

$$K_{\mathcal{S}}^a = msk_1^a \sum_{j \in \mathcal{S}_a} P_{B+1-j} = \gamma_1^a \sum_{j \in \mathcal{S}_a} P_{B+1-j}$$

Finally, output

$$K_{\mathcal{S}} = (K_{\mathcal{S}}^1, \cdots, K_{\mathcal{S}}^A)$$

Note that the the aggregate key now consists of A group elements.

Broadcast$(param, K_{\mathcal{S}}, \hat{\mathcal{S}}, PK, bsk)$: The aggregate key $K_{\mathcal{S}} = (K_{\mathcal{S}}^1, \cdots, K_{\mathcal{S}}^A)$ is broadcast to all users in $\hat{\mathcal{S}}$ as follows. For the subset of user ids $\hat{\mathcal{S}}$ and $1 \le \hat{a} \le \hat{A}$, define

$$\hat{\mathcal{S}}_{\hat{a}} = \{\hat{i} \bmod B + 1 | \hat{i} \in \hat{\mathcal{S}}, \lceil \hat{i}/B \rceil = \hat{a}\}$$

Randomly choose $\hat{t} \in \mathbb{G}_q$ and set for $1 \le \hat{a} \le \hat{A}$

$$b_{\hat{\mathcal{S}}_{\hat{a}}} = \sum_{\hat{j} \in \hat{\mathcal{S}}_{\hat{a}}} Q_{B+1-\hat{j}}$$

Output the broadcast aggregate key as:

$$K_{(\mathcal{S},\hat{\mathcal{S}})} = (\hat{t}Q, \mathcal{K}_1, \mathcal{K}_2)$$

where

$$\begin{aligned}\mathcal{K}_1 &= \left(\hat{t}\left(PK_4^1 + b_{\hat{\mathcal{S}}_1}\right), \cdots, \hat{t}\left(PK_4^{\hat{A}} + b_{\hat{\mathcal{S}}_{\hat{A}}}\right)\right)\\ \mathcal{K}_2 &= \left(\{e(P_B, \hat{t}Q_1) \cdot e(K_{\mathcal{S}}^a, Q)^{bsk}\}_{1 \le a \le A}\right)\end{aligned}$$

Note that $K_{(\mathcal{S},\hat{\mathcal{S}})}$ now comprises of $\mathcal{O}(A + \hat{A})$ group elements.

Decrypt$(param, \mathcal{C}, K_{(\mathcal{S},\hat{\mathcal{S}})}, i, \hat{i}, d_{\hat{i}}, \mathcal{S}, \hat{\mathcal{S}})$: If $i \notin \mathcal{S}$ or $\hat{i} \notin \hat{\mathcal{S}}$, output $\perp$. Otherwise, compute $a = \lceil i/B \rceil$ and $b = i \bmod B + 1$ and set:

$$\begin{aligned}a_{\mathcal{S}_a} &= \sum_{j \in \mathcal{S}_a, j \neq b} P_{B+1-j+b}\\ b_{\mathcal{S}_a} &= \sum_{j \in \mathcal{S}_a} P_{B+1-j}\end{aligned}$$

Also, compute $\hat{a} = \lceil \hat{i}/B \rceil$ and $\hat{b} = \hat{i} \bmod B + 1$ and set:

$$a_{\hat{\mathcal{S}}_{\hat{a}}} = \sum_{\hat{j} \in \hat{\mathcal{S}}_{\hat{a}}, \hat{j} \neq \hat{b}} P_{B+1-\hat{j}+\hat{b}}$$

Let $\mathcal{C} = (c_0, c_1, c_2, c_3)$ and

$$K_{(\mathcal{S},\hat{\mathcal{S}})} = \left(\hat{k}_0, \left(\hat{k}_1^1, \cdots, \hat{k}_1^{\hat{A}}\right), \left(\hat{k}_2^1, \cdots, \hat{k}_2^A\right)\right)$$

Return the decrypted message $\hat{\mathcal{M}}$ as:

$$\hat{\mathcal{M}} = c_3 \cdot \hat{k}_2^a \cdot \left(\frac{e(b_{\mathcal{S}_a}, c_1)\, e(a_{\mathcal{S}_a}, c_0)}{e(b_{\mathcal{S}_a}, c_2)}\right) \cdot \left(\frac{e\left(d_{\hat{i}} + a_{\hat{\mathcal{S}}_{\hat{a}}}, \hat{k}_0\right)}{e\left(P_{\hat{b}}, \hat{k}_1^{\hat{a}}\right)}\right)$$

Note that the aforementioned framework is for a single data owner. For m' data owners, m' such frameworks will need to be instantiated. The proof of correctness for this construction is similar to that for the basic KAC construction and is hence avoided.

6.2 Semantic Security

For the non-adaptive CPA security of the generalized extended KAC construction, we state the following theorem.

Theorem 3. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be bilinear elliptic curve subgroups of prime order q. For any triple of positive integers (n, m, B) such that $B \le n, m$, the generalized extended KAC handling n data classes and m users is (τ, ϵ, B) CPA secure if the asymmetric decision (τ, ϵ, B, B)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$.*

The proof of this theorem is similar to that for Theorem 1. Finally we note that the extended KAC construction may also be extended using techniques similar to those in Section 4 to achieve CCA security.

6.3 Performance and Efficiency

The choice of A, A' and B play an important role in system performance. As is clear from the construction, the ciphertext always consists of a constant number of group elements. The public parameter comprises of $\mathcal{O}(B)$ group elements, while the broadcast aggregate key $K_{(\mathcal{S},\hat{\mathcal{S}})}$ as well as the public key PK consist of $\mathcal{O}(A + \hat{A})$ group elements each. Thus choosing a smaller value of B is useful for applications requiring low overhead aggregate keys. Table 2 summarizes the theoretical space complexities for the generalized scheme in the order notation. Note that any group element in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ is assumed to have $\mathcal{O}(\eta_1)$, $\mathcal{O}(\eta_2)$ and $\mathcal{O}(\eta_T)$ space complexity respectively.

TABLE 2: Space Complexities for Various Components : Generalized Extended KAC

Component	Space Complexity
$param$	$\mathcal{O}(B(\eta_1+\eta_2))$
msk	$\mathcal{O}((A+\hat{A})\log q)$
PK	$\mathcal{O}(A\eta_1+\hat{A}\eta_2)$
bsk	$\mathcal{O}(\log q)$
$\mathcal{C}$	$\mathcal{O}(\eta_2+\eta_T)$
$d_{\hat{i}}$	$\mathcal{O}(\eta_2)$
$K_{\mathcal{S}}$	$\mathcal{O}(A\eta_1)$
$K_{(\mathcal{S},\hat{\mathcal{S}})}$	$\mathcal{O}(\left(\hat{A}+1\right)\eta_2+A\eta_T)$

TABLE 3: Timing Results for Primitive Operations

Primitive Operation	Time Taken (in seconds)
Point Addition	7.01×10^{-6}
Scalar Multiplication	6.56×10^{-4}
Group Multiplication	9.80×10^{-6}
Bilinear Pairing	2.40×10^{-2}

TABLE 4: Inter-VM Communication over the Network

Type of Element	Size (bytes)	Transfer Time (milliseconds)
$\mathbb{G}_1$	40	0.137
$\mathbb{G}_2$	80	0.210
$\mathbb{G}_T$	240	0.291

6.4 Addition and Revocation in Multi-User Environments

Addition of new users and revocation of existing ones is of particular significance in multi-user environments where the number of users, as well as user access rights are expected to change dynamically. We note here that addition of new users is easily handled in our extended KAC construction by generating a new key for each newly added user in the system, and ensuring that all future aggregate key broadcast operations take into account the access rights of the new users in addition to the already existing ones. Also, observe that addition of new users does not require the system to re-generate the core building block of size $B \times B$ in the extended construction. It is only the parameter $\hat{A}$ that increases on the addition of new users, which is essentially equivalent to instantiating more instances of the same core building block. It follows that neither the public parameters nor the existing owner/user keys need to be updated in any way. Thus combination with broadcast encryption ensures that the system scales to larger number of users efficiently.

User revocation is also achieved without any additional cost since it is built into the very definition of broadcast encryption systems. Suppose that at a certain point of time, a data owner wishes to revoke the access of a particular data user to all future documents that she puts on the cloud. She simply asks the system to exclude the identity of this data user from all future broadcast operations, implying that the excluded data user can no longer access aggregate keys corresponding to any data that the data owner puts on the cloud in future. Clearly, this requires no extra cost, and the corresponding security guarantee follows from the collusion resistance property of the extended KAC construction. In particular, since any aggregate key that the revoked user had access to does not cover the indices of any of the future documents, the collusion resistance property ensures that the knowledge of prior aggregate keys does not compromise the knowledge of these new documents. Of course, the revoked user continues to have its designated access to the previously existing documents, but she could have downloaded and saved them prior to revocation anyway; so revoking user access to existing documents seems superfluous. In case the data owner wishes to modify an existing document, she could use a different index for it to ensure that a revoked user cannot access it in the future.

Revocation is often associated with the presence of rogue/compromised users that threaten the security of a system. Tracing such users is, however, more involved and is best handled using *traitor-tracing* systems [34]. Assume that a data owner broadcasts the aggregate key for a certain subset $\mathcal{S}$ of plaintext messages to a set $\hat{\mathcal{S}}$ of legitimate users. The risk for the data owner is that a malicious agent could hack a user in this recipient set $\hat{\mathcal{S}}$, extract its secret key and build a publicly available decoder that allows anyone outside the intended recipient set to extract the plaintext content for themselves. This is where a traitor tracing system comes handy because it allows a data owner to run a tracing algorithm that interacts with the publicly available malicious decoder to identify at least one compromised user index, whose access can then be revoked. A detailed description of how a traitor tracing scheme may be efficiently combined with the extended KAC framework is outside the scope of the current work. We point out, however, that there exist several propositions in the literature for combining traitor-tracing with broadcast encryption [35], [36] with varying efficiency that could be used in the extended KAC framework.

.

7 Experimental Results on the Cloud

This section presents an experimental validation of the performance and efficiency of the extended KAC construction with broadcast aggregate key in a public cloud based set-up consisting of three VMs - a data owner client VM that performed the **Encrypt** operations, the data user client VM that performed the **Decrypt** operation, and a trusted third party server VM that performed the rest of the operations. Each of the two client VMs were equipped with 1GB RAM each, while the server VM was equipped with 4GB RAM since it performed the bulk of the computational operations. There have been several efficient software implementations for bilinear pairings reported in the literature [37], [38] although the corresponding implementation details are not always available. In our implementation, we use the Pairing-Based Cryptography [39] library, which is an open-source C library, and provides efficient APIs for asymmetric prime order bilinear pairings and elliptic curve operations over a "Type-F" curve, which belongs to the family of Barreto-Naehrig Curves [40]. TCP-based network communication between the VMs was achieved via standard BSD socket APIs provided by the `socket` module in python. In response to each successful inter VM transfer from the server to the client, the client responded with a single byte of acknowledgement.

We present a case study for an extended KAC system that supports $n = 1000$ documents and $m = 1000$ users. First, we assume the simplest possible configuration where only a single instance of the core $B \times B$ block is instantiated for $B = 1000$. Our case study also samples two random subsets $\mathcal{S}$ and $\hat{\mathcal{S}}$ of size 500 and 500 respectively. This essentially means that the system is broadcasting the aggregate key for 500 documents to 500 users. We first enumerate in Table 3 the time taken for each primitive function - namely point addition in $\mathbb{G}_1$ and $\mathbb{G}_2$, scalar multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$, group operations in $\mathbb{G}_T$, and bilinear pairing computations. We do not present the timing for exponentiations in $\mathbb{G}_T$, since all such operations are generally in conjunction with pairing, and are therefore simulated via a scalar multiplication in either $\mathbb{G}_1$ or $\mathbb{G}_2$, followed by the application of a pairing. Also, quite evidently, the timing overhead for pairings is much larger than most other operations and is hence expected to dominate the time required for various KAC algorithms. In Table 5 we analyze the expected time complexity for the various algorithms in extended KAC in this scenario, based on the number of primitive operations in each algorithm. We then compare it with the actual time required for each algorithm in our simulated framework for extended KAC. We point out that the additional overhead over the expected time required for each algorithm may be attributed to network delays and the time required to serialize various group elements for input and output. The inter-VM communication specifics for a single element in each of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, along with their sizes, are presented separately in Table 4. This excludes a standard RTT of 29 microseconds for receipt of the acknowledgement from the client.

We next evaluate the memory requirement of the system for the same case study with 1000 documents and 1000 users, using different combination of values for A, A' and B in order to highlight the trade-off resulting from various choices of these system parameters. The results are presented in Table 6. We note that the memory requirement results are in accordance with the theoretical expectations. With increase in B, $param$ takes greater memory, while the memory requirements for msk, PK, $K_{\mathcal{S}}$ and $K_{(\mathcal{S},\hat{\mathcal{S}})}$ increase.

For a better understanding of Table 6, consider the sixth row. This row corresponds to a setting of the extended KAC system for $n = 1000$ documents and $m = 1000$ users, with the aggregate key for 500 documents being broadcast to 500 users. The core block is of size 100×100 (that is, $B = 100$) and we instantiate 10×10 instances of this core block (that is, $A = \hat{A} = 10$). The public parameter $param$ in this case consists of 200 elements each from $\mathbb{G}_1$ and $\mathbb{G}_2$ and hence has size 24 KB. The master secret key msk consists of 20 random elements from $\mathbb{Z}_q$ and consequently has size 400 bytes. The public key PK consists of 20 elements each from $\mathbb{G}_1$ and $\mathbb{G}_2$, and has size 2.4 KB. The aggregate key $K_{\mathcal{S}}$ consists of 10 elements from $\mathbb{G}_1$ and hence has size 400 bytes. The broadcast aggregate key $K_{(\mathcal{S},\hat{\mathcal{S}})}$ consists of 11 group elements from $\mathbb{G}_2$ and 10 group elements from $\mathbb{G}_T$, and hence has size 2.08 KB. Finally, the length of the ciphertext is not mentioned in the table because it is constant at 480 bytes for all combinations of $\left(B, A, \hat{A}\right)$. Refer Table 2 and Table 4 for checking the number of elements and size of each element respectively in the above discussion. We also point out here that the size of $K_{\mathcal{S}}$ and $K_{(\mathcal{S},\hat{\mathcal{S}})}$ are independent of the size of the target document and user subsets $\mathcal{S}$ and $\hat{\mathcal{S}}$ respectively.

TABLE 6: Memory Requirement Results: Trade-off over various choices of B, A and $\hat{A}$

B	A	$\hat{A}$	$param$ (in bytes)	msk (in bytes)	PK (in bytes)	$K_{\mathcal{S}}$ (in bytes)	$K_{(\mathcal{S},\hat{\mathcal{S}})}$ (in bytes)
1	1000	1000	240	80000	480000	80000	200080
2	500	500	480	40000	240000	40000	100080
5	200	200	1200	8000	48000	8000	40080
10	100	100	2400	4000	24000	4000	20080
50	20	20	12000	800	4800	800	4080
100	10	10	24000	400	2400	400	2080
500	2	2	120000	80	480	80	480
1000	1	1	240000	40	240	40	280

8 Conclusions and Discussions

In this paper, we have proposed an efficiently implementable version of the basic key-aggregate cryptosystem (KAC) in [6] with low overhead ciphertexts and aggregate keys, using asymmetric bilinear pairings. Our construction serves as an efficient solution for several data sharing applications on the cloud, including collaborative data sharing, product license distribution and medical data sharing. We have proved our construction to be fully collusion resistant and semantically secure against a non-adaptive adversary under appropriate security assumptions. We have then demonstrated how this construction may be modified to achieve CCA-secure construction, which is, to the best of our knowledge, the first CCA secure KAC construction in the cryptographic literature. We have further demonstrated how the basic KAC framework may be efficiently extended and generalized for securely broadcasting the aggregate key among multiple data users in a real-life data sharing environment. This provides a crucial pathway in designing a scalable fully public-key based online data sharing scheme for large-scale deployment on the cloud. We have presented simulation results to validate the space and time complexity requirements for our scheme. The results establish that KAC with aggregate key broadcast outperforms other existing secure data sharing schemes in terms of performance and scalability.

References

[1] IDC Enterprise Panel. It cloud services user survey, pt. 3: What users want from cloud services providers, august 2008.

[2] Sherman SM Chow, Yi-Jun He, Lucas CK Hui, and Siu Ming Yiu. Spice–simple privacy-preserving identity-management for cloud environment. In *Applied Cryptography and Network Security*, pages 526–543. Springer, 2012.

[3] Cong Wang, Sherman S.-M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for secure cloud storage. Cryptology ePrint Archive, Report 2009/579, 2009. http://eprint.iacr.org/.

[4] Sherman SM Chow, Cheng-Kang Chu, Xinyi Huang, Jianying Zhou, and Robert H Deng. Dynamic secure cloud storage with provenance. In *Cryptography and Security: From Theory to Applications*, pages 442–464. Springer, 2012.

[5] Erik C Shallman. Up in the air: Clarifying cloud storage protections. *Intell. Prop. L. Bull.*, 19:49, 2014.

[6] Cheng-Kang Chu, Sherman SM Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. *Parallel and Distributed Systems, IEEE Transactions on*, 25(2):468–477, 2014.

[7] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005*, pages 258–275. Springer, 2005.

TABLE 5: Timing Results: Expected v/s Actual for 1000 documents and 1000 users with subset sizes of 500 each

	SetUp	**OwnerKeyGen**	**OwnerEncrypt**	**SystemEncrypt**
Point Addition	0	0	1	1
Scalar Multiplication	3998	4	4	2
Group Multiplication	0	0	1	0
Bilinear Pairing	0	0	1	0
Expected Timing (in s)	2.62	2.62×10^{-4}	2.66×10^{-2}	1.32×10^{-3}
Actual Timing (in s)	5.61	5.90×10^{-4}	2.67×10^{-2}	1.76×10^{-3}
Difference (Actual-Estimated) (in s)	2.99	3.28×10^{-4}	9.49×10^{-4}	4.38×10^{-4}

(a) Timing Results: Part-1

	UserKeyGen	**Extract**	**Broadcast**	**Decrypt**
Point Addition	0	500	500	1500
Scalar Multiplication	1	1	4	0
Group Multiplication	0	0	0	3
Bilinear Pairing	0	0	2	5
Expected Timing (in s)	6.56×10^{-4}	4.16×10^{-3}	5.42×10^{-2}	1.31×10^{-1}
Actual Timing (in s)	1.00×10^{-3}	1.11×10^{-2}	6.38×10^{-2}	1.49×10^{-1}
Difference (Actual-Estimated) (in s)	3.47×10^{-4}	6.91×10^{-3}	9.52×10^{-3}	1.81×10^{-2}

(b) Timing Results: Part-2

[8] Selim G Akl and Peter D Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems (TOCS)*, 1(3):239–248, 1983.

[9] Gerald C Chick and Stafford E Tavares. Flexible access control with master keys. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 316–322. Springer, 1990.

[10] Wen-Guey Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *Knowledge and Data Engineering, IEEE Transactions on*, 14(1):182–188, 2002.

[11] Giuseppe Ateniese, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Provably-secure time-bound hierarchical key assignment schemes. *Journal of cryptology*, 25(2):243–270, 2012.

[12] Ravinderpal S Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, 27(2):95–98, 1988.

[13] Yan Sun and KJ Liu. Scalable hierarchical access control in secure group communications. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1296–1306. IEEE, 2004.

[14] William C King and Bjorn Hjelm. Centralized key management, March 24 2015. US Patent 8,990,555.

[15] Mikhail J Atallah, Marina Blanton, Nelly Fazio, and Keith B Frikken. Dynamic and efficient key management for access hierarchies. *ACM Transactions on Information and System Security (TISSEC)*, 12(3):18, 2009.

[16] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *Advances in CryptologyEUROCRYPT 2002*, pages 466–481. Springer, 2002.

[17] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456. Springer, 2005.

[18] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005*, pages 114–127. Springer, 2005.

[19] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology-CRYPTO 2006*, pages 290–307. Springer, 2006.

[20] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, pages 47–53. Springer, 1985.

[21] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[22] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and coding*, pages 360–363. Springer, 2001.

[23] Fuchun Guo, Yi Mu, and Zhide Chen. Identity-based encryption: how to decrypt multiple ciphertexts using a single decryption key. In *Pairing-Based Cryptography–Pairing 2007*, pages 392–406. Springer, 2007.

[24] Fuchun Guo, Yi Mu, Zhide Chen, and Li Xu. Multi-identity single-key decryption without random oracles. In *Information Security and Cryptology*, pages 384–398. Springer, 2008.

[25] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

[26] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[27] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE, 2007.

[28] Melissa Chase. Multi-authority attribute based encryption. In *Theory of cryptography*, pages 515–534. Springer, 2007.

[29] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *Parallel and Distributed Systems, IEEE Transactions on*, 24(1):131–143, 2013.

[30] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 103–114. ACM, 2009.

[31] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology-Eurocrypt 2004*, pages 207–222. Springer, 2004.

[32] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in CryptologyEurocrypt 2002*, pages 45–64. Springer, 2002.

[33] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *Advances in Cryptology-CRYPTO 2009*, pages 671–689. Springer, 2009.

[34] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Trans. Information Theory*, 46(3):893–910, 2000.

[35] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 573–592, 2006.

[36] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 501–510, 2008.

[37] Chitchanok Chuengsatiansup, Michael Naehrig, Pance Ribarski, and Peter Schwabe. Panda: Pairings and arithmetic. In *Pairing-Based Cryptography - Pairing 2013 - 6th International Conference, Beijing, China, November 22-24, 2013, Revised Selected Papers*, pages 229–250, 2013.

[38] Eric Zavattoni, Luis J. Dominguez Perez, Shigeo Mitsunari, Ana H. Sánchez-Ramírez, Tadanori Teruya, and Francisco Rodríguez-Henríquez. Software implementation of an attribute-based encryption scheme. *IEEE Trans. Computers*, 64(5):1429–1441, 2015.

[39] Ben Lynn. The Pairing-Based Cryptography Library.

[40] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, pages 319–331. Springer, 2006.