Scalable Key-Aggregate Cryptosystem for Secure 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. In this paper, we address this problem by proposing a dynamic, scalable and efficient key aggregate encryption scheme with provable security and user revocation properties. We lay special focus on how the scheme can be actually deployed on the cloud for multiple data owners and data users. We present simulation results to prove the efficiency of the scheme and compare its performance with other existing cryptosystems for online data sharing in literature.

Index Terms—Cloud Computing, Data Sharing, Data Security, Key-Aggregate Cryptosystem, Provable Security, Scalability, Revocation

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share and exchange data, while saving the pangs of manual data exchanges, and avoiding the creation of redundant or out-of-date documents.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 smpartphone 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

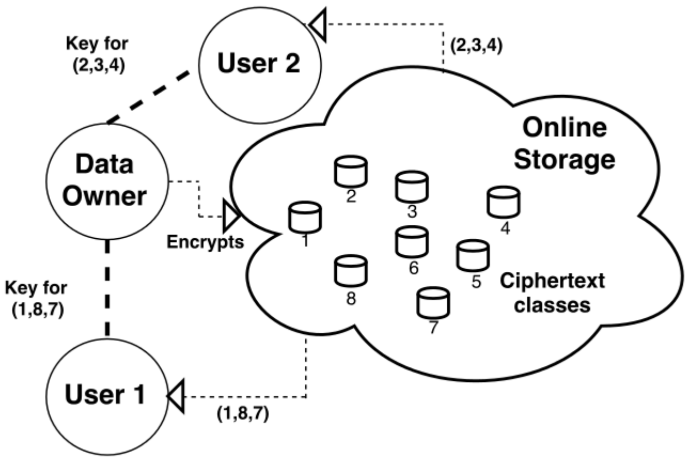

Fig. 1: Example of Online Data Sharing

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 the data owner is using an online data sharing service such as Microsoft OneDrive [5] to store her research documents. She wishes to add an additional layer of security for her data by storing them in an encrypted fashion. Now, she intends to share specific subsets of these documents with different collaborators. For this, she needs to provide each of her collaborators with decryption rights to specific classes of the data that they are authorized to access. She might also wish to dynamically update the delegated access rights based on changes to the data/credibility issues. The challenge therefore is to provide her with a secure and efficient online *partial* data sharing scheme that allows updates to user access rights on the fly.

A näive (and extremely inefficient) solution is to have a different decryption key for each ciphertext class, and share them accordingly with 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. Not many of the present cryptosystems seem to address this issue of reducing the decryption key size for online data sharing schemes, as we describe next.

1.1 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. One of the most popular techniques for access control in online data storage is to use a pre-defined hierarchy of secret keys 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 [6], [7], [8], [9], [10], [7]. 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. Compact key encryption for the symmetric key setting has been used in [11] to solve the problem of concisely transmitting large number of keys in the broadcast scenario. However, symmetric key sharing via a secured channel is costly and not always practically viable for many applications on the cloud. Proxy re-encryption is another technique to achieve fine-grained access control and scalable user revocation in unreliable clouds [12], [13]. However, proxy re-encryption essentially transfers the responsibility for secure key storage from the delegatee to the proxy and is susceptible to collusion attacks. It is also important to ensure that the transformation key of the proxy is well protected, and every decryption would require a separate interaction with the proxy, which is inconvenient for applications on the cloud.

1.2 The Key-Aggregate Encryption Scheme

The most efficient proposition pertaining to our problem statement, to the best of our knowledge, is made in [14]. 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 by Boneh *et.al.* [15] that allows broadcasting of data (encrypted by the same public key) among multiple users, each of whom possess their own private keys for decryption. In KAC, when a user demands for a particular subset of the available classes of data, the data owner computes an aggregate key which integrates the power of the individual decryption keys corresponding to each class of data. However, KAC as proposed in [14] 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 [14]. Formal proofs are considered to be of paramount importance in establishing the security of any cryptographic scheme, since they define the specific adversarial models against which the scheme is secure.
2) Secondly, the scheme proposed in [14] 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 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.

1.3 Our Contributions

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

1) In this paper we propose an efficient key-aggregate cryptosystem (KAC) for online data sharing on the cloud using efficiently implementable asymmetric bilinear pairings.
2) We further generalize the basic KAC construction using a two-tier scheme and demonstrate how this allows multiple data owners to share their data while enjoying full data privacy.
3) We demonstrate how the basic KAC framework may be efficiently extended for securely broadcasting the aggregate key among multiple data users in a real-life data sharing environment. This in turn allows us to build a fully public-key based online data sharing scheme that is highly scalable.
4) All our constructions are fully collusion resistant and are proven to be statically CPA and CCA secure under different complexity assumptions.
5) Implementation details and simulation results demonstrate that our proposed construction actually scales better than traditional hierarchical cryptosystems in terms of space and time complexity requirements.

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 [14] 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 $\mathcal{S} \subset \{1, 2, \cdots, n\}$. 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 the aggregate key $K_\mathcal{S}$ corresponding to the subset $\mathcal{S}$ such that $i \in \mathcal{S}$. Outputs the decrypted message.

2.2 Extension to Basic KAC : Broadcasting the Aggregate Key among Multiple Users

A major limitation of the basic KAC framework described in Section 2.1 is that it does not address how the aggregate key may be distributed among multiple data users in a real-life data sharing environment. The authors of [14] suggest using secure channels for key distribution. This is, however, not a very practical suggestion in view of the fact that the number of data users in the system may be arbitrarily large. We present here an extended framework that combines the basic KAC framework with public-key based broadcast encryption systems [15] to build a full-fledged public key based online data sharing scheme. The extended framework is parameterized by the number of users m along with the number of data classes n. In addition to producing a single aggregate key for an arbitrarily large subset S of data classes, this scheme also provides a way to securely broadcast it to an authorized subset $\hat{\mathcal{S}} \subseteq \{1, \cdots, m\}$ of users.

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) **Encrypt**$(param, PK, bsk, i, \mathcal{M})$: Takes as input a data class $i \in \{1, \cdots, n\}$ and the plaintext data $\mathcal{M}$. Outputs the corresponding ciphertext $\mathcal{C}$.
4) **UserKeyGen**$(param, msk, \hat{i})$: Takes as input the data user id $\hat{i} \in \{1, \cdots, m\}$ and outputs the corresponding secret key $d_{\hat{i}}$.
5) **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.
6) **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 $i \in \mathcal{S}$.
7) **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.

2.3 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 ciphertext classes, as input. The game proceeds through the following stages:

1) **Init**: Algorithm $\mathcal{A}$ begins by outputting a set $\mathcal{S}^* \subset \{1, 2, \cdots, n\}$ of data classes that it wishes to attack. Challenger $\mathcal{B}$ randomly chooses a ciphertext 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 ciphertext 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 **Decrypt**$(\mathcal{C}, j, \mathcal{S}, K_{\mathcal{S}})$ for some subset $\mathcal{S} \subseteq \{1, \cdots, n\}$ such that $j \in \mathcal{S}$.
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 ciphertext 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 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.

2.4 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 2.2 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 $\mathcal{S}^* \subset \{1, 2, \cdots, n\}$ of data classes and a set $\hat{\mathcal{S}}^* \subset \{1, 2, \cdots, m\}$ of users that it wishes to attack. Challenger $\mathcal{B}$ randomly chooses a ciphertext 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 ciphertext 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 earlier.

2.5 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 ma[if it satisfies the following properties:

- Bilinear: For all $P_1 \in \mathbb{G}_1, Q_1 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_{\shortparallel}$, 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 $\hat{e'}(P, Q) \neq 1$
- Computability: There exists an efficient algorithm to compute $\hat{e'}(P_1, Q_1) \forall P_1 \in \mathbb{G}_1, Q_1 \in \mathbb{G}_2$

From a cryptographic standpoint, pairings can be broadly classified into three categories as follows (for more details refer [16]):

- *Type-1 symmetric pairings*: In such pairings $\mathbb{G}_1 = \mathbb{G}_2$. These pairings are defined exlusively over supersingular curves and require subgroups of extremely large order.
- *Type-2 asymmetric pairings*: In such pairings $\mathbb{G}_1 \neq \mathbb{G}_2$, but there exists an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.
- *Type-3 asymmetric pairings*: In such pairings $\mathbb{G}_1 \neq \mathbb{G}_2$, and there exists no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

The case where $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists efficiently computable homomorphisms in either direction is equivalent to type one pairings. There also exist Type-4 pairings over non-cyclic subgroups [16], but they are not pertinent to our discussions on KAC. The original proposal for KAC in [14] is based on Type-1 symmetric pairings. However, from a practical deployment perspective on the cloud, Type-1 pairings are usually not efficient enough to be deployed due to their specific curve requirements and large group size demands. So, in this paper, we focus on the use of efficiently implementable Type-2 and Type-3 pairings such as the Tate pairing [17], the Eta pairing [18], the Ate pairing [19] and even advanced pairings that are extremely efficient such as the R-Ate pairing [19]. We next state the complexity assumptions that allow us to formally prove the security of our proposed KAC schemes based on such pairings.

2.6 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. Finally, we assume the existence of an efficiently computable asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Finally, for two independent randomly chosen $\alpha_1, \alpha_2 \in \mathbb{Z}_q$, we denote by R_x the term $\alpha_1^x R$ and by $\hat{R}_x$ the term $\alpha_2^x R$.

2.7 The Complexity Assumptions

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 [15]:

- *The asymmetric decision l-BDHE problem*: Given an input $(I = (H, P, Q, Y_{P,\alpha,l}, Y_{Q,\alpha,l}), Z)$, 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 asymmetric decision (l_1, l_2)-BDHE problem*: Given an input $((I_1, I_2), (Z_1, Z_2))$, where $I_1 = (H_1, P, Q, Y_{P,\alpha_1,l_1}, Y_{Q,\alpha_1,l_1})$ and $I_2 = (H_2, P, Q, Y_{P,\alpha_2,l_2}, Y_{Q,\alpha_2,l_2})$, and the bilinear pairing e, decide if $(Z_1, Z_2) = \Big(e(P_{l_1+1}, H_1), e(\hat{P}_{l_2+1}, H_2)\Big)$.

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

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

where the probability is over random choice of $H \in \mathbb{G}_2$, $T \in \mathbb{G}_T$ and $\alpha \in \mathbb{Z}_q$, 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 asymmetric (l_1, l_2)-BDHE and outputs a decision bit $b \in \{0,1\}$. We say that $\mathcal{B}$ has advantage ϵ in solving the asymmetric decision (l_1, l_2)-BDHE problem if

$$
\begin{aligned}
|Pr[\mathcal{B}\left((I_1, I_2), \left(e(P_{l_1+1}, H_1), e(\hat{P}_{l_2+1}, H_2)\right)\right) = 0] \\
-Pr[\mathcal{B}((I_1, I_2), (T_1, T_2)) = 0]| \geq \epsilon
\end{aligned}
$$

where the probability is over random choice of $H_1, H_2 \in \mathbb{G}_2$, $T_1, T_2 \in \mathbb{G}_T$ and $\alpha_1, \alpha_2 \in \mathbb{Z}_q$, 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 asymmetric decision $(\tau, \epsilon, l_1, l_2)$-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_1, l_2)-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$.*

3 A Basic KAC Using Asymmetric Bilinear Pairings

In this section, we present the design of the basic key-aggregate cryptosystem. Our construction is based on asymmetric bilinear pairings that are practical and efficiently implementable and do not require non-singular elliptic curves. In this respect, our const For the basic case, we assume a single data owner who stores her data in n different classes, with no hierarchical organization within each class. In later discussions, we show how the system can be scaled to multiple users, and also to allow hierarchical constructions within each class. Our scheme ensures that the ciphertext and aggregate key are of constant size, while the public parameter size is linear in the number of data classes n. We prove the scheme to non-adaptively CPA secure under the asymmetric l-BDHE exponent assumption.

3.1 Construction

We present the basic construction of our proposed KAC. 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

$$\mathcal{C} = (tQ, t(PK_2 + Q_i), \mathcal{M}.e(P_n, tQ_1))$$

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

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

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:

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

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

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

3.2 Performance and Efficiency

The decryption time for any subset of ciphertext classes $\mathcal{S}$ is essentially dominated by the computation of $W_{\mathcal{S}} = \sum_{j \in \mathcal{S}} P_{n+1-j+i}$. However, if a user has already computed $\sum_{j \in \mathcal{S'}} P_{n+1-j+i}$ for a subset S' similar to S, then she can easily compute the desired value by at most $|\mathcal{S} - \mathcal{S'}|$ operations. For similar subsets S and S', this value is expected to be fairly small. As suggested in [15], for subsets of very large size$(n - r, r \ll n)$, an advantageous approach could be to pre-compute $\sum_{j=1}^{j=n} P_{n+1-j+i}$ corresponding to $i = 1$ to n, which would allow the user to decrypt using only r group operations, and would require only r elements of $param$. Similar optimizations would also hold for the encryption operation where pre-computation of $\sum_{j=1}^{j=n} P_{n+1-j}$ is useful for large subsets.

3.3 Semantic Security of the Basic KAC

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 pair of positive integers $n', n(n' > n)$, the basic KAC 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 $(I = (H, P, Q, Y_{P,\alpha,n}, Y_{Q,\alpha,n}), Z)$ (where 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 ciphertext classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a ciphertext class $i \in \mathcal{S}$.

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

- $param$ 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 $\gamma P - P_i$ and $\gamma Q - Q_i$ respectively, for some γ randomly chosen from $\mathbb{Z}_q$. Note that this is equivalent to setting msk as $\gamma - \alpha^i$.

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

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

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

$$K_{\overline{\mathcal{S}}} = \sum_{j \notin \mathcal{S}} \alpha^{n+1-j} PK_1$$

as desired. Moreover, $\mathcal{B}$ 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 values are 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 $\mathcal{M}_1$ from the set of possible plaintext messages in class i, and gives them to $\mathcal{B}$. To generate the challenge, $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge 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

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

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 $\mathcal{L}'_{\text{BDHE}}$, $|Pr[\mathcal{B}(I, Z)] - \frac{1}{2}| = |Adv_{\mathcal{A},n'} - \frac{1}{2}| \geq \epsilon$. 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 does not require the use of random oracles. □

4 Extended KAC with Aggregate Key Broadcast

In this section, we present a construction for the extended KAC with public-key based aggregate key broadcast introduced in 2.2. Our construction efficiently combines the basic KAC instance presented in 3.1 with the public key based broadcast encryption system presented in [15] to handle n data classes and m users .

SetUp$(1^\lambda, n, m)$: Randomly pick $\alpha_1, \alpha_2 \in \mathbb{Z}_q$. Output the system parameter as

$$param = (P, Q, Y_{P,\alpha_1,n}, Y_{Q,\alpha_1,n}, Y_{P,\alpha_2,m}, Y_{Q,\alpha_2,m})$$

Discard α_1 and α_2. Note that in the forthcoming discussion, $P_i = \alpha_1^i P$ and $\hat{P}_i = \alpha_2^i P$. Similar definitions follow for Q_i and $\hat{Q}_i$ respectively.

OwnerKeyGen(): Randomly pick $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{Z}_q$. Let $msk_1 = \gamma_1$ and $msk_2 = \gamma_2$. Set the master secret key msk to (msk_1, msk_2). Let $PK_1 = \gamma_1 P$, $PK_2 = \gamma_1 Q$, $PK_3 = \gamma_2 P$ and $PK_4 = \gamma_2 Q$. Set the public key $PK = (PK_1, PK_2, Pk_3, PK_4)$. Finally set the broadcast secret key $bsk = \gamma_3$. Output (msk, PK, bsk).

Encrypt$(param, PK, bsk, i, \mathcal{M})$: 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:

$$\mathcal{C} = (tQ, (t - bsk)PK_2, t(PK_2 + Q_i), \mathcal{M}.e(P_n, tQ_1))$$

UserKeyGen$(param, msk, \hat{i})$: Output the private key for user with id $\hat{i}$ as:

$$d_{\hat{i}} = msk_2 \hat{P}_{\hat{i}}$$

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

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

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

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

Broadcast$(param, K_{\mathcal{S}}, \hat{\mathcal{S}}, PK, bsk)$: Broadcasts the aggregate key $K_{\mathcal{S}}$ to all users in $\hat{\mathcal{S}}$ as follows. Randomly choose $\hat{t} \in \mathbb{G}_q$ and set

$$\hat{b}_{\hat{\mathcal{S}}} = \sum_{\hat{j} \in \hat{\mathcal{S}}} \hat{Q}_{m+1-\hat{j}}$$

Output

$$K_{(\mathcal{S}, \hat{\mathcal{S}})} = (\hat{t}Q, PK_4 + \hat{b}_{\hat{\mathcal{S}}}, \mathcal{K})$$

where

$$\mathcal{K} = \left(e(\hat{P}_m, \hat{Q}_1).e(K_{\mathcal{S}}, Q)^{bsk} \right)$$

Note that the actual group element corresponding to $K_{\mathcal{S}}$ is difficult to recover from $\mathcal{K}$. However, as we demonstrate next, this knowledge is not explicitly necessary for decryption.

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, set

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

Let $\mathcal{C} = (c_0, c_1, c_2, c_3)$ and $K_{(\mathcal{S},\hat{\mathcal{S}})} = (\hat{k}_0, \hat{k}_1, \hat{k}_2)$. Output the decrypted message as

$$\hat{\mathcal{M}} = c_3.\hat{k}_2.\left(\frac{e(b_{\mathcal{S}}, c_1)e(a_{\mathcal{S}}, c_0)}{e(b_{\mathcal{S}}, c_2)}\right).\left(\frac{e(d_{\hat{i}} + a_{\hat{\mathcal{S}}}, \hat{k}_0)}{e(\hat{P}_{\hat{i}}, \hat{k}_1)}\right)$$

The proof of correctness of the extended KAC scheme is similar to that for the basic construction and is avoided. We now prove static security of the extended scheme in the CPA model.

4.1 Semantic Security of the Extended KAC

We state the following theorem.

Theorem 2. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be bilinear elliptic curve subgroups of prime order q. For any pair of positive integers $n', n(n' > n)$ and another pair of positive integers $m', m(m' > m)$, the extended KAC is (τ, ϵ, n', m') CPA secure if the asymmetric decision (τ, ϵ, n, m)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$.*

Proof: Let $\mathcal{A}$ be a τ-time adversary such that $|Adv_{\mathcal{A},n',m'} - \frac{1}{2}| > \epsilon$ for a KAC system parameterized with a given pair (n, m). We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the asymmetric (n, m)-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$. Algorithm $\mathcal{B}$ takes as input a random asymmetric (n, m)-BDHE challenge $((I_1, I_2), (Z_1, Z_2))$ (where $I_1 = (H_1, P, Q, Y_{P,\alpha_1,l_1}, Y_{Q,\alpha_1,l_1})$, $I_2 = (H_2, P, Q, Y_{P,\alpha_2,l_2}, Y_{Q,\alpha_2,l_2})$ and (Z_1, Z_2) is either $\Big(e(P_{n+1}, H_1), e(\hat{P}_{m+1}, H_2)\Big)$ 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 data classes and the set $\hat{\mathcal{S}}$ of data users that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a ciphertext class $i \in \mathcal{S}$.

SetUp: $\mathcal{B}$ sets up the extended KAC system by generating the following:

- $param$ is set as

  $$param = (P, Q, Y_{P,\alpha_1,n}, Y_{Q,\alpha_1,n}, Y_{P,\alpha_2,m}, Y_{Q,\alpha_2,m})$$

  and supplied to $\mathcal{A}$.
- $\mathcal{B}$ randomly chooses $\gamma_1, \gamma_2 \in \mathbb{Z}_q$ and computes the following:

  $$\begin{aligned} PK_1 &= \gamma_1 P - P_i \\ PK_2 &= \gamma_1 Q - Q_i \\ PK_3 &= \gamma_2 P - \sum_{\hat{j}\in\hat{\mathcal{S}},\hat{j}\neq\hat{i}} \hat{P}_{m+1-\hat{j}} \\ PK_4 &= \gamma_2 Q - \sum_{\hat{j}\in\hat{\mathcal{S}},\hat{j}\neq\hat{i}} \hat{Q}_{m+1-\hat{j}} \end{aligned}$$

  $\mathcal{B}$ sets the public key $PK = (PK_1, PK_2, PK_3, PK_4)$ and supplies it to $\mathcal{A}$. Note that this is equivalent to setting the master secret key msk as:

  $$(msk_1, msk_2) = \left(\gamma_1 - \alpha_1^i, \gamma_2 - \sum_{\hat{j}\in\hat{\mathcal{S}},\hat{j}\neq\hat{i}} \alpha_2^{m+1-\hat{j}+\hat{i}}\right)$$

  Finally, $\mathcal{B}$ chooses a random $\gamma_3 \in \mathbb{Z}_q$ and implicitly sets the value of the secret broadcast key bsk to $t - \gamma_3$, such that $H_1 = tQ$. We see later how this implicit definition manifests in the actual game.

$\mathcal{B}$ also supplies $\mathcal{A}$ with all the secret keys for all users not in $\hat{\mathcal{S}}$. In particular, the key for user $\hat{i}$ is set as:

$$d_{\hat{i}} = \gamma_2 \hat{P}_{\hat{i}} - \sum_{\hat{j}\in\hat{\mathcal{S}},\hat{j}\neq\hat{i}} \hat{P}_{m+1-\hat{j}+\hat{i}}$$

Observe that $d_{\hat{i}} = \alpha^{\hat{i}} PK_3$ as desired. In addition, $\mathcal{A}$ is also supplied with the collusion aggregate key $K_{(\overline{\mathcal{S}},\hat{\mathcal{S}})}$. For this, $\mathcal{B}$ first computes

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

and sets

$$\overline{\mathcal{K}} = Z_2.e(K_{\overline{\mathcal{S}}}, H_1 - \gamma_3 Q)$$

Note that the implicit definition of bsk is used here. Finally, $\mathcal{B}$ provides $\mathcal{A}$ with the aggregate key:

$$K_{(\overline{\mathcal{S}},\hat{\mathcal{S}})} = (H_2, \gamma_2 H_2, \overline{\mathcal{K}})$$

It can be easily shown that whenever $Z_2 = e(\hat{P}_{l_2+1}, H_2)$, this is a valid collusion aggregate key for the sets $\mathcal{S}$ of data classes and $\hat{\mathcal{S}}$ of user ids.

Challenge: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages in class i, and gives them to $\mathcal{B}$. To generate the challenge, $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge as $(\mathcal{C}, \mathcal{M}_0, \mathcal{M}_1)$, where

$$\mathcal{C} = (H_1, PK_1, \gamma_1 H_1, \mathcal{M}_b.Z_1)$$

Once again, it can be easily shown that when $Z_1 = e(P_{n+1}, H_1)$, then $(\mathcal{C}, \mathcal{M}_0, \mathcal{M}_1)$ is a valid challenge to $\mathcal{A}$ as in a real attack.

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

We conclude that $\mathcal{B}$ has at least the same advantage ϵ as $\mathcal{A}$ in solving the asymmetric (n, m)-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$. This concludes the proof of Theorem 1. Note that this proof is again in the standard model and does not require the use of random oracles. $\square$

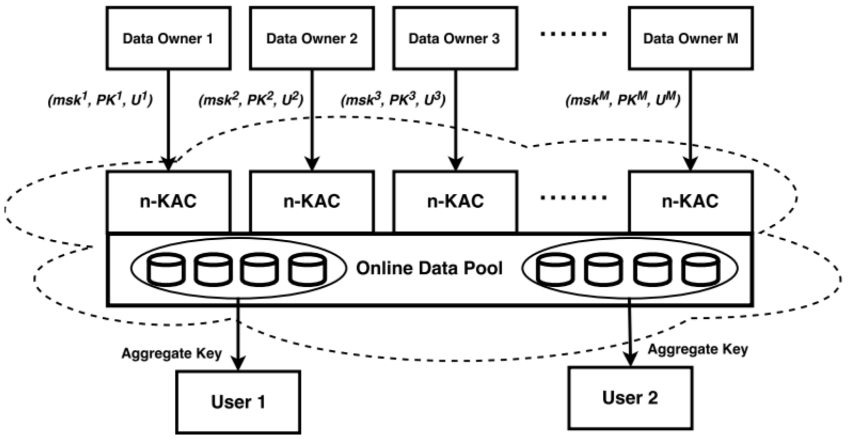

Fig. 2: Generalized Multi-Data Owner KAC

4.2 Data Privacy

In any online data sharing environment with multiple data owners, data privacy is an essential requirement. In particular, the aggregate key supplied by one data owner should not leak information about another data owner to an unauthorized user. This problem however does not arise in our construction if a new parallel instance of the KAC construction in Section 4 is run for each data owner. Each instance can handle n data classes and can cater to m data users. In order to distinguish between data classes belonging to different instances, each data class is assigned a double index (i_1, i_2), where i_1 is the instance/owner index, and i_2 is the class index specific to the instance. Each instance i_1 is characterized by its own master secret key msk^{i_1}, public key PK^{i_1} and broadcast key bsk^{i_1}. The main advantages of this multi-user setup are as follows:

1) All the parallel instances can share the same public $param$, which needs to be setup exactly once by the system administrator.
2) Data privacy is ensured for each individual data owner. In particular, the aggregate key issued by one data owner does not compromise the security of some other owner's data.
3) The system is scalable since it allows new users to register and share their data without having to alter the underlying basic KAC set-up in any way. This also implies that the basic security assumption of the system is not affected by the increase in the number of data classes.

Also note that the number of unique ordered tuples $\left(msk^{i_1}, PK^{i_1}, dsk^{i_1}\right)$ is q^3 for the extended construction, meaning that a single setup can support $O(q^3)$ data owners. Finally, if a data owner wishes to store more than n classes of data or cater to more than n data users, she may be allocated more than one instance of the KAC construction in Section 4. Figure 2 illustrates a practical data sharing scenario with multiple owners sharing their data and distributing access to different users. In summary, the KAC construction is an ideal choice for building a fully public-key based online data sharing scheme.

4.3 Embedding Plaintext Messages in Bilinear Groups

The two KAC constructions - basic and extended presented in Sections 3 and 4 respectively, assume that all plaintext messages $\mathcal{M}$ may be efficiently embedded as elements in the multiplicative group G_T. Unfortunately, embedding any general class of data such as multimedia as elements of a bilinear group is extremely challenging. However, a workaround may be readily proposed. We first note that in any ciphertext output by **Encrypt**, the message $\mathcal{M}$ is essentially multiplied with a random secret group element ρ. Rather than embedding $\mathcal{M}$ as a group element, we propose hashing ρ using a collision resistant hash function H, and then outputting $\mathcal{M} \odot H(\rho)$ in the ciphertext (here $\odot$ denotes an appropriate operator). In order to ensure that the constructions are still provably secure in the standard model, we propose that H be chosen from the family of *smooth projective hash functions* [20], that do not require the use of random oracles to prove security. Smooth projective hash functions are very efficient to construct and can be designed to be collision-resistant [21], making them an ideal choice for our constructions.

Finally, we note that our KAC constructions are agnostic of the manner in which the data owner organizes her data. In particular, our construction is easily adaptable for hierarchical data structures, since a data owner could create an aggregate key corresponding to all the data classes rooted at any internal node, and then broadcast it to the target user group.

5 Simulation Studies : Performance and Efficiency of KAC

6 Applications of KAC

The key-aggregate encryption systems described in this paper are primarily meant for data sharing on the cloud. In this section, we point out some specific applications in which KAC proves to be a very efficient solution.

Online Collaborative Data Sharing

The foremost application of KAC is in secure data sharing for collaborative applications. Applications such as Google

Drive [22] and Dropbox [23] allow users to share their data on the cloud and delegate access rights to multiple users to specific subsets of their whole data. Even government and corporate organizations require secure data sharing mechanisms for their daily operations. KAC can be easily set up to function on top of standard data sharing applications to provide security and flexibility. Data classes may be viewed as folders containing similar files. The fact that our proposed KAC is identity based means that each folder can have its own unique ID chosen by the data owner. Also, the fact that the ciphertext overhead is only logarithmic in the number of data classes implies that space requirement for any data owner is optimal. Finally, the aggregate key also has low overhead and can be transmitted via a secure channel such as a password protected mail service. Since KAC is easily extensible to multiple data owners, the system is practically deployable for a practical data sharing environment. The other advantage of KAC is that once a system is setup with a set of multilinear maps and public parameters, the same setup with the same set of public parameters can be reused by multiple teams within the same organization. Since data owned by each individual owner is insulated from access by users who do not have the corresponding aggregate key, and each data owner has her own tuple of public, private and authentication keys, a single KAC can support multiple data sharing units, while guaranteeing the same underlying security. This saves the cost of setting up new multilinear maps and public parameters each time.

Distribution of Product License and/or Activation Keys

Suppose a company owns a number of products, and intends to distribute the license files (or activation keys) corresponding to these to different users. The KAC framework allows them to put these keys on the cloud in an encrypted fashion, and distribute an aggregate key corresponding to the license files for multiple products to legally authenticated customers as per their requirements. The legal authentication comes from the fact the user who buys multiple products from the company is given the authentication key and the aggregate key that allows her to decrypt the license file for each product. Since both these keys are of constant size, distributing these to users is easier than providing a separate license file to each user.

Patient controlled encryption (PCE)

Patient controlled encryption (PCE) is a recent concept that has been studied in the literature [11]. PCE allows a patient to upload her own medical data on the cloud and delegate decryption rights to healthcare personnel as per her requirement. KAC acts as an efficient solution to this problem by allowing patients to define their own hierarchy of medical data and delegate decryption rights to this data to different specialists/medical institutions using aggregate keys in an efficient fashion. Given the multitude of sensitive digital health records existent in today's world, storing this data in local/personal machines is not a viable solution and the cloud seems the best alternative. KAC thus provides a two-way advantage in this regard. Not only does it allow people from across the globe to store their health data efficiently and safely, but also allows them to envisage the support of expert medical care from across the globe.

7 A Generalized Basic KAC Construction

In this section, we present a a generalized basic KAC construction. The essential idea is to run n_A parallel instances of the basic scheme parameterized by n_B number of classes, so that the overall system can handle as many as $n = n_A \times n_B$ data classes. Each of these instances share the same set of public parameters, but use their own set of private and public keys. This leads to a trade-off between the overhead for various system parameters.

7.1 The Construction

We present the construction for generalized KAC in this section. The construction uses techniques similar to the generalized broadcast encryption scheme proposed in [15]. The generalization in [15] trades off the ciphertext size with the public key size. Our scheme, on the other hand, trades off the public parameter size with the public key size and the aggregate key size, while still maintaining constant ciphertext overhead.

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

KeyGen(n_A): Randomly pick $\gamma_1, \cdots, \gamma_{n_A} \in \mathbb{Z}_q$. Let $msk_j = \gamma_j$, $PK_j^1 = \gamma_j P$ and $PK_j^2 = \gamma_j Q$ for $1 \leq j \leq n_A$. Set the master secret key $msk = (msk_1, \cdots, msk_{n_A})$. Set $PK^1 = (PK_1^1, \cdots, PK_{n_A}^1)$ and $PK^2 = (PK_1^2, \cdots, PK_{n_A}^2)$. Finally, set the public key $PK = (PK^1, PK^2)$ and output the tuple (msk, PK).

Encrypt$(PK, i, \mathcal{M})$: Compute $a_i = \lceil i/n_B \rceil$ and $b_i = i \bmod B$. Randomly choose $t \in \mathbb{Z}_q$ and output the ciphertext $\mathcal{C}$ as

$$\mathcal{C} = (c_1, c_2, c_3) = (tQ, t(PK_a^2 + Q_b), \mathcal{M}.e(P_n, tQ_1))$$

Extract$(msk, \mathcal{S})$: For the subset of class indices $\mathcal{S}$ and $1 \leq a \leq n_A$, define

$$\begin{aligned} \mathcal{S}_a &= \{i | i \in \mathcal{S}, \lceil i/n_B \rceil = a\} \\ \mathcal{S}'_a &= \{b_i = i \bmod n_B | i \in \mathcal{S}_a\} \end{aligned}$$

Next, for $1 \leq a \leq n_A$, compute

$$K_{\mathcal{S}}^a = msk_a \sum_{b \in \mathcal{S}'_a} P_{n+1-b}$$

Note that this is indirectly equivalent to setting $K_{\mathcal{S}}^a$ to $\sum_{b \in \mathcal{S}'_a} \alpha^{n+1-b} PK_a^1$. Finally, output

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

Note that the the aggregate key now consists of n_A elements.

Decrypt$(\mathcal{C}, i, K_{\mathcal{S}}, \mathcal{S})$: If $i \notin \mathcal{S}$, output $\perp$. Otherwise, compute $a_i = \lceil i/n_B \rceil$ and $b_i = i \bmod B$ and set:

$$\begin{aligned} A_{\mathcal{S}} &= \sum_{(b \in \mathcal{S}'_{a_i}, b \neq b_i} P_{n+1-b_i+b} \\ B_{\mathcal{S}} &= \sum_{(b \in \mathcal{S}'_{a_i}} P_{n+1-b} \end{aligned}$$

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

$$\hat{\mathcal{M}} = c_3.\frac{e(K_{\mathcal{S}}^{a_i} + A_{\mathcal{S}}, c_1)}{e(B_{\mathcal{S}}, c_2)}$$

Correctness of the algorithm may be easily proved similarly as in Section 3.1. Finally, we note that setting $n_A = 1$ and $n_B = n$ gives the basic construction of Section 3.1.

7.2 Performance and Efficiency

The choice of n_A and n_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 n_B group elements, while the aggregate key as well as the public key consist of n_A group elements each. Thus choosing a smaller value of n_B is useful for applications requiring low overhead aggregate keys. However, choosing $n_A = n_B = \sqrt{n}$ minimizes overall system overhead combining all parameters.

We present the simulation studies with different values of n_A and n_B here.

7.3 Semantic Security of General KAC

For the non-adaptive CPA security of the generalized 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 pair of positive integers $n', n (n' > n)$, the generalized KAC is (τ, ϵ, n') CPA secure if the asymmetric decision (τ, ϵ, n)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$.*

The proof of this theorem is very similar to the proof of Theorem 1 and is hence avoided.

8 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 proposed in the cryptographic literature.

8.1 Additional Requirements for CCA Security

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

- A signature scheme $(SigKeyGen, Sign, Verify)$.
- 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.

8.2 Construction

We will demonstrate in the following section that the security of CCA-secure Dynamic KAC for n data classes is based on the asymmetric $(n + 1)$-BDHE assumption, instead of the asymmetric n-BDHE assumption as for the basic scheme. For consistency of notation, we describe here the CCA-secure dynamic KAC for $n - 1$ 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(): Same as in the basic scheme of Section 3.1.

Encrypt$(PK, i, \mathcal{M})$: 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

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

Output the ciphertext $\mathcal{C}$.

Extract$(msk, \mathcal{S})$: Same as in Section 3.1.

Decrypt$(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}})$: 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, pick a random $w \in \mathbb{Z}_q$ and 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, 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

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

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

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

where x is chosen uniformly from $\mathbb{Z}_q$. To verify this claim, set $x = w + \sum_{j \in \mathcal{S}} \alpha^{n+1-j}$. Moreover, since 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 on the ciphertext class i for the message m to be decrypted.

8.3 Proof of CCA-Security

We begin by recalling 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 ϵ. For a more complete description, refer [24].

Theorem 4. *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 is $(\tau, \epsilon_1 + \epsilon_2, n-1, q_D)$ CCA-secure if the decision (τ, ϵ, n, n)-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ and the signature scheme is $(t, \epsilon_2, 1)$ 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 $(I = (H, P, Q, Y_{P,\alpha,l}, Y_{Q,\alpha,l}), Z)$ (where 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 ciphertext classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a ciphertext class $i \in \mathcal{S}^*$.

SetUp: $\mathcal{B}$ should generate the public $param$, public key PK, the authentication key U, and the aggregate key $K_{\overline{\mathcal{S}^*}}$, and provide them to $\mathcal{A}$. Algorithm $\mathcal{B}$ 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 $\gamma P - V^*_{SIG} P_n - P_i$ and $\gamma Q - V^*_{SIG} Q_n - Q_i$ respectively 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

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

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: Algorithm $\mathcal{A}$ now issues decryption queries. Let $(j, \mathcal{C})$ be a decryption query where $j \in \mathcal{S}^*$. Let $\mathcal{C} = ((c_0, c_1, c_2), \sigma, V_{SIG})$. Algorithm $\mathcal{B}$ 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. Otherwise, the challenger picks a random $x \in \mathbb{Z}_q$. It then sets

$$\begin{aligned}
\hat{h}_0 &= (V_{SIG} - V^*_{SIG}) P_n + P_j - P_i \\
\hat{h}'_0 &= (V_{SIG} - V^*_{SIG})^{-1} (P_{j+1} - P_{i+1}) \\
\hat{h}_2 &= xP + (V_{SIG} - V^*_{SIG})^{-1} P_1 \\
\hat{h}_1 &= u\hat{h}_2 + xd_0 + d'_0
\end{aligned}$$

$\mathcal{B}$ responds with $K = c_2 \frac{e(\hat{h}_1, c_0)}{e(\hat{h}_2, c_1)}$. To see that this response is as in a real attack game, set $x' = x + \alpha(V_{SIG} - V^*_{SIG})^{-1}$ and observe that $\hat{h}_2 = x'P$ and $\hat{h}_1 = x'(PK_1 + P_i + V_{SIG} P_n) - P_{n+1}$. Furthermore, since x is uniform in $\mathbb{Z}_q$, x' is also uniform in $\mathbb{Z}_q$. Thus, $\mathcal{B}$'s response is a valid decryption as required.

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

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

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

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

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 $(\mathbb{G}_1, \mathbb{G}_2)$.

We now bound the probability that $\mathcal{B}$ aborts the simulation as a result of 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 u 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 4. □

The extended KAC construction for multiple users presented in Section 4, as well the generalized basic KAC construction from Section 7 may also be similarly modified to obtain CCA security.

9 CONCLUSIONS

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] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[13] Qin Liu, Chiu C Tan, Jie Wu, and Guojun Wang. Reliable re-encryption in unreliable clouds. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5. IEEE, 2011.

[14] 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.

[15] 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.

[16] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[17] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *Information Theory, IEEE Transactions on*, 45(5):1717–1719, 1999.

[18] Florian Hess, Nigel P Smart, and Frederik Vercauteren. The eta pairing revisited. *Information Theory, IEEE Transactions on*, 52(10):4595–4602, 2006.

[19] Jean-Luc Beuchat, Jorge E González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In *Pairing-Based Cryptography-Pairing 2010*, pages 21–39. Springer, 2010.

[20] 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.

[21] 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.

[22] Ian Paul. Google Drive: The Pros and Cons.

[23] CloudPro. Dropbox goes big on security with Enterprise offering.

[24] 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.