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

Dr.K.Srinivas Babu*1  N.Sindhu*2  Dr.K.Rameshwaraiah*3
*1, *2 Professor Department of Computer Science and Engineering  *3 M.Tech

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 m0 data owners while reducing the reducing the secure channel requirement from O(mm0) in the standalone case to O(m + m0).


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share and exchange data, while saving the pangs of manual data exchanges, and avoiding the creation of redundant or out-of-date documents.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

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 cryptography (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 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 cryptography (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 $O(m + m_0)$ for $m$ data users and $m_0$ data owners, which is an improvement over the $O(m^m m_0)$ 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 \( m_0 \) 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.

**II. 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 issue security of these schemes.

**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 \( i \in S \). 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 \( \lambda \). Outputs the public parameter \( \text{param} \).
2) **KeyGen():** Outputs the public key \( P \_K \) and the master-secret key \( msk \) for a data owner registering in the system.
3) **Encrypt(param, P \_K, i, M):** Takes as input the public key parameter \( P \_K \), the data class \( i \) and the plaintext data \( M \). Outputs the corresponding ciphertext \( C \).
4) **Extract(param, msk, S):** Takes as input the master secret key \( msk \) and a subset of data classes \( S \subseteq \{1, 2, \ldots, n\} \). Computes the aggregate key \( KS \) for all encrypted messages belonging to these subset of classes.
5) **Decrypt(param, C, i, S, KS):** Takes as input the ciphertext \( C \), the data class \( i \) and the aggregate key \( KS \) corresponding to the subset \( S \) such that \( i \in S \). 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 \( A \) and a challenger \( B \), both of whom are given \( n \), the total number of message classes, as input. The game proceeds through the following stages:

1) **Init:** Algorithm \( A \) begins by outputting a set \( S \ast \subseteq \{1, 2, \ldots, n\} \) of data classes that it wishes to attack. Challenger \( B \) randomly chooses a message class \( i \in S^* \).
2) **SetUp:** Challenger \( B \) sets up the KAC system by generating the public parameter \( \text{param} \), the public key \( P \_K \) and the master secret key \( msk \). Since collusion attacks are allowed in our framework, \( B \) furnishes \( A \) with the aggregate key \( KS^* \) that allows \( A \) to decrypt any message class \( j \in S \ast \).
3) **Query Phase 1:**

**Algorithm A** adaptively issues decryption queries \( q_1, \ldots, q_v \) where a decryption query comprises of the tuple \( (j, C) \), where \( j \in S \ast \). The challenger responds with a valid decryption of \( C \).

4) **Challenge:** A picks at random two messages \( M_0 \) and \( M_1 \) from the set of possible plaintext messages belonging to class \( i \) and provides them to \( B \). To generate the challenge, \( B \) randomly picks \( b \in \{0, 1\} \) and sets the challenge to \( A \) as \( (C^b, M_0, M_1) \), where \( C^b = \text{Encrypt}(\text{param}, P \_K, i, Mb) \).

5) **Query Phase 2:** Algorithm \( A \) continues to adaptively issue decryption queries \( q_{v+1}, \ldots, \).
qQD where a decryption query now comprises of the tuple (j, C) under the restriction that C = C* . The challenger responds as in phase 1. 6) Guess: The adversary A outputs a guess b0 of b. If b0 = b, 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 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 A is non-adaptive; it chooses S, and obtains the aggregate decryption key for all message classes outside of S, before it even sees the public parameters param or the public key PK. Let AdvA,n denote the probability that 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, , n, qQD ) CCA secure if for all non-adaptive τ-time algorithms A that can make a total of qQD decryption queries, we have that |Adv − 1| < .

CPA Security: We say that a key-aggregate encryption system is (τ, , n) CPA secure if it is (τ, , 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 G1 and G2 be two (additive) cyclic elliptic curve subgroups of the same prime order q. Let GT 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 G1 and G2 respectively. A mapping e : G1 × G2 −→ GT is said to be a bilinear map if it satisfies the following properties:

- Bilinear: For all P1 ∈ G1 , Q1 ∈ G2 , and a, b ∈ Zq , we have e(aP1 , bQ1 ) = e(P1 , Q1 )ab
- Non-degeneracy: If P and Q be the generators for G1 and G2 respectively where neither group only contains the point at infinity, then e(P, Q) = 1.
- Computability: There exists an efficient algorithm to compute e(P1 , Q1 ) for all P1 ∈ G1 and Q1 ∈ G2.

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 G1 and G2, along with their generators P and Q. We also assume the existence of a multiplicative cyclic group GT, also of order q with identity element 1. Let α be a randomly chosen element in Zq . For any point R in either G1 or G2, let Rx = αx R, where x is an integer. We denote by YP,α,l the set of 2l−1 points (R1 , R2 , • • • , Rl+2 , • • • , R2l ). Note that the term Rl+1 is missing. We assume the existence of an efficiently computable asymmetric bilinear pairing e : G1 × G2 −→ GT . Finally, any group element in G1, G2 or GT is assumed to have size O(η1 ), O(η2 ) and O(ηT ) 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 The asymmetric decision 1-BDHE problem: Given an input (H, I = (P, Q, YP,α,l , YQ,α,l ), Z ), where H ∈ G2 and Z ∈ GT , and the bilinear pairing e, decide if Z = e(P+l , H ). The extended asymmetric decision 1-BDHE problem: Given an input ((H1 , H2 ), I = (P, Q, YP,α,l , YQ,α,l )\{Z1 , Z2 \}), and the bilinear pairing e, where H1 , H2 ∈ G2 and Z1 , Z2 ∈ GT decide if (Z1 , Z2 ) = (e(P+l , H1 ), e(P+l , H2 )). Let A be a τ-time algorithm that takes an input challenge for asymmetric 1-BDHE and outputs a decision bit b ∈ {0, 1}. We say that A has advantage in solving the asymmetric decision 1-BDHE problem if |Pr[A(H, I , e(P+l , H ) ) = 0] − Pr[A(I , Z ) = 0]| ≥ where the probability is over random choice of H ∈ G2 , random choice of Z ∈ GT , random choice of α ∈ Zq , and random bits used by A. We refer to the distribution on the left as LBDHE and the distribution on the right as RBDHE.

III. A PROVABLY CURVE BASIC KAC USING ASYMERIC 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 S ⊆ {1, • • • , 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 data owner using asymmetric bilinear pairings. As mentioned in Section 2, we assume the existence of equi-prime order (for a $\lambda$-bit prime $q$) elliptic curve subgroups $G_1$ and $G_2$, along with their generators $P$ and $Q$. We also assume the existence of a multiplicative cyclic group $GT$, also of order $q$ with identity element $1$. Finally, we assume there exists an asymmetric bilinear pairing $e: G_1 \times G_2 \rightarrow GT$. The notations used in the forthcoming discussion are already introduced in Section 2.

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

$\text{KeyGen}()$: Randomly pick $\gamma \in \mathbb{Z}_q$. Set the master secret key $\text{msk}$ to $\gamma$. Let $PK_1 = \gamma P$ and $PK_2 = \gamma Q$. Set the public key $PK = (PK_1, PK_2)$. Output (msk, PK). $\text{Encrypt}((\text{param}, PK, i, M))$: For a message $M \in GT$ belonging to class $i \in \{1, 2, \ldots, n\}$, randomly choose $t \in \mathbb{Z}_q$. Output the ciphertext $C$ as $C = (c_0, c_1, c_2) = (tQ, t(PK_2 + Qi), M \cdot e(P, tQ))$.

### 3.2 Correctness

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

$M = c_2, e(K, P) = e(P_{n+1}^{i-j_1} - P_{n+1}^{i-j_1}, c_0)$

$= c_2, e(P_{n+1}^{i-j_1}, c_0)$

$= c_2, e(P_{n+1}^{i-j_1}, tPK_2)$

$= c_2, c_0, e(P_{n+1}^{i-j_1}, tQ)$

$= c_2, e(P_{n+1}, tQ) \cdot e(Q_{n+1, j}, tQ)$

$= M, e(P_{n+1}, tQ)$

$\Rightarrow M$.

### 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 $G_1$ and $G_2$ be bilinear elliptic curve subgroups of prime order $q$. For any positive integer $n$, the basic KAC handling $n$ data classes is (1, $n$)-CPA secure if the asymmetric decision (1, $n$)-BDHE assumption holds in ($G_1, G_2$).

**Proof.** Let $A$ be an $t$-time adversary such that $|\text{Adv} - 1| >$ for a KAC system parameterized with a given $n$. We build an algorithm $B$ that has advantage at least in solving the asymmetric $n$-BDHE problem in ($G_1, G_2$). Algorithm $B$ 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 $e(P, tQ)$ or a random value in $GT$).

### IV. CONCLUSION

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


