Dynamic Key-Aggregate Cryptosystem on Elliptic Curves for Online Data Sharing

Sikhar Patranabis(✉), Yash Shrivastava, and Debdeep Mukhopadhyay

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur, Kharagpur, India
{sikhar.patranabis,yash.shrivastava,debdeep}@cse.iitkgp.ernet.in

Abstract. The recent advent of cloud computing and the IoT has made it imperative to have efficient and secure cryptographic schemes for online data sharing. 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 with appropriate credentials. An efficient and recently proposed solution in this regard is to use the concept of aggregation that allows users to decrypt multiple classes of data using a single key of constant size. In this paper, we propose a secure and dynamic key aggregate encryption scheme for online data sharing that operates on elliptic curve subgroups while allowing dynamic revocation of user access rights. We augment this basic construction to a generalized two-level hierarchical structure that achieves optimal space and time complexities, and also efficiently accommodates extension of data classes. Finally, we propose an extension to the generalized scheme that allows use of efficiently computable bilinear pairings for encryption and decryption operations. Each scheme is formally proven to be semantically secure. Practical experiments have been conducted to validate all claims made in the paper.

Keywords: Key-aggregate cryptoystem · Online data sharing · Semantic security · Dynamic access rights

1 Introduction

The advent of cloud computing and the Internet of Things (IoT) has led to a massive rise in the demand for online data storage and data sharing services. Two very important paradigms that any data sharing service provider must ensure are privacy and flexibility. Since online data almost always resides in shared environments (for instance, multiple virtual machines running on the same physical device), ensuring privacy is a non trivial task. Current technology for secure data sharing comes in two major flavors - trusting a third party auditor [1] or using the user's own key to encrypt her data [2]. Figure 1 describes a realistic online data sharing set-up. Suppose a data owner stores multiple classes of encrypted data online with the intention of providing users decryption keys to one or more such ciphertext classes, based on their respective credentials. She might also wish to dynamically update the delegated access rights based on

A. Biryukov and V. Goyal (Eds.): INDOCRYPT 2015, LNCS 9462, pp. 25–44, 2015.
DOI: 10.1007/978-3-319-26617-6_2

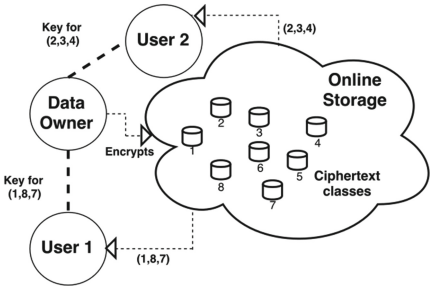

Fig. 1. Example of online data sharing

changes to the data/credibility issues. The challenge therefore is to provide her with a secure and efficient online 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. A more efficient proposition is the key-aggregate encryption (KAC) scheme proposed in [3] that combines the power of individual decryption keys, for ciphertext classes in a given subset, into a single key for that subset. This key is specific to the designated subset, meaning that it cannot be used to decrypt any ciphertext class outside that subset. KAC derives its roots from the seminal work by Boneh *et.al.* [4] that allows broadcasting of data (encrypted by the same public key) among multiple users, each of whom possess their own private keys for decryption. Both these schemes make use of bilinear mappings on multiplicative cyclic groups.

Contributions: In this paper, we propose a basic key-aggregate scheme on additive elliptic subgroups that delegate decryption rights to multiple ciphertext classes using a single constant sized key. The scheme is dynamic in nature, that is, it allows the data owner to revoke access rights of users without having to change the entire set-up, unlike in the existing KAC scheme. We then generalize this scheme into a two-level construction that allows flexible public key extension and maintains constant ciphertext size, while avoiding many of the pitfalls of earlier hierarchical schemes. We provide a formal proof of semantic security for the generalized scheme. We further extend the generalized scheme to allow using popular and efficiently implementable elliptic curve pairing schemes. We compare the time and space requirements of the proposed generalized scheme under various operating configurations. We also compare the performance of our proposed scheme, in terms of key size and resource utilization, with that of other existing schemes in literature.

Organization: The rest of the paper is organized as follows. Section 2 provides a brief overview of state of the art data sharing schemes. Section 3 introduces the notion of key aggregate cryptosystem, and provides a description of the complexity assumptions used to prove the semantic security of our proposed schemes. Our basic dynamic key-aggregate scheme is presented in Sect. 4. We follow up

with a more generalized two-tiered construction of the scheme for efficient public key extension in Sect. 5, and prove its semantic security. A further extension for the generalized scheme that allows using efficiently implementable pairings is introduced and proved semantically secure in Sect. 6. Experimental results using Tate pairings based implementations of the extended scheme are presented in Sect. 7. Finally Sect. 8 concludes the paper.

2 Related Work

In this section we present a brief overview of public and private key cryptographic schemes in literature for secure online data sharing. While many of them focus on key aggregation in some form or the other, very few have the ability to provide constant size keys to decrypt an arbitrary number of encrypted entities. One of the most popular techniques for access control in online data storage is to use a pre-defined hierarchy of secret keys [5] 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. 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 [6] 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 [7]. 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.

The authors of [3] proposes an efficient scheme, namely KAC, that allows secure and efficient sharing of data on the cloud. The scheme is a public-key cryptosystem that uses constant size ciphertexts such that efficient delegation of decryption rights for any set of ciphertexts are possible. 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. KAC as proposed in [3] suffers from three major drawbacks, each of which we address in this paper. First of all, the security assumption of KAC seems to be the Bilinear Diffie Hellman Exponent (BDHE) assumption [8]; however no concrete proofs of semantic security are provided by the authors in [3]. Secondly, with respect to user access rights, KAC is a static scheme in the sense that once a user is in possession of the aggregate key corresponding to a subset of files from data owner, the owner cannot dynamically revoke the permission of the client for accessing one or more updated files. Since dynamic changes in access rights is extremely common in online data storage, this

scenario needs to be tackled. Finally, the public key extension of KAC proposed in [3] is extremely cumbersome and resource consuming since registration of each new public key-private key pair requires the number of classes to be extended by the original number of classes.

3 Preliminaries

We begin by formally defining the Key Aggregate Cryptosystem (KAC), and stating the complexity assumptions used to prove the security of the encryption schemes proposed in this paper.

3.1 The Key Aggregate Cryptosystem (KAC)

A key aggregate cryptosystem is an ensemble of the following randomized algorithms:

1. **Setup**$(1^\lambda, n)$: Takes as input the number of ciphertext classes n and the group order parameter λ. Outputs the public parameter PK. Also computes a secret parameter t used for encryption which is not made public. It is only known to data owners with credentials to control client access rights.
2. **Keygen**(): Outputs the public and master-secret key pair: $(PK = \gamma P, msk = \gamma)$.
3. **Encrypt**(PK, i, m): Takes as input the public key parameter PK, the ciphertext class i and the message m. Outputs the ciphertext $\mathcal{C}$ corresponding to the message m belonging to class i.
4. **Extract**$(msk = \gamma, \mathcal{S})$: Takes as input the master secret key γ and a subset $\mathcal{S} \subset \{1, 2, \cdots, n\}$. Computes the aggregate key $K_\mathcal{S}$ and the dynamic access control parameter U. The tuple $(K_\mathcal{S}, U)$ is transmitted via a secure channel to users that have access rights to $\mathcal{S}$.
5. **Decrypt**$(K_\mathcal{S}, U, \mathcal{S}, i, \mathcal{C} = \{c_1, c_2, c_3\})$: Takes as input the aggregate key $K_\mathcal{S}$ corresponding to a subset $\mathcal{S} \subset \{1, 2, \cdots, n\}$, the dynamic access parameter U, the ciphertext class i and the ciphertext $\mathcal{C}$. Outputs the decrypted message m.

3.2 Semantic Security of KAC

We now define the semantic security of a key-aggregate encryption system against an adversary using the following game between an attack algorithm $\mathcal{A}$ and a challenger $\mathcal{B}$. Both $\mathcal{A}$ and $\mathcal{B}$ 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 receivers that it wishes to attack. For each ciphertext class $i \in \mathcal{S}$, challenger $\mathcal{B}$ performs the **SetUp-i**, **Challenge-i** and **Guess-i** steps. Note that the number of iterations is polynomial in $|S|$.

2. **SetUp-i**: Challenger $\mathcal{B}$ generates the public $param$, public key PK, the access parameter U, and provides them to $\mathcal{A}$. In addition, $\mathcal{B}$ also generates and 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. **Challenge-i**: Challenger $\mathcal{B}$ performs an encryption of the secret message m_i belonging to the i^{th} class to obtain the ciphertext $\mathcal{C}$. Next, $\mathcal{B}$ picks a random $b \in (0, 1)$. It sets $K_b = m_i$ and picks a random K_{1-b} from the set of possible plaintext messages. It then gives $(\mathcal{C}, K_0, K_1)$ to algorithm $\mathcal{A}$ as a challenge.
4. **Guess-i**: The adversary $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{A}$ wins and the challenger $\mathcal{B}$ loses. Otherwise, the game moves on to the next ciphertext class in $\mathcal{S}$ until all ciphertext classes in $\mathcal{S}$ are exhausted.

If the adversary $\mathcal{A}$ fails to predict correctly for all ciphertext classes in $\mathcal{S}$, only then $\mathcal{A}$ loses the game. Let $AdvKAC_{\mathcal{A},n}$ denote the probability that $\mathcal{A}$ wins the game when the challenger is given n as input. We say that a key-aggregate encryption system is (τ, ϵ, n) semantically secure if for all τ-time algorithms $\mathcal{A}$ we have that $|AdvKAC_{\mathcal{A},n} - \frac{1}{2}| < \epsilon$ where ϵ is a very small quantity. 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.

3.3 The Complexity Assumptions

We now introduce the complexity assumptions used in this paper. In this section, we make several references to bilinear non-degenerate mappings on elliptic curve sub-groups, popularly known in literature as pairings. For a detailed descriptions on pairings and their properties, refer [9].

The First Complexity Assumption: Our first complexity assumption is the l-BDHE problem [4] in a bilinear elliptic curve subgroup $\mathbb{G}$, defined as follows. Given a vector of $2l+1$ elements $(H, P, \alpha P, \alpha^2 P, \cdots, \alpha^l P, \alpha^{l+2} P \cdots, \alpha^{2l} P) \in \mathbb{G}^{2l+1}$ as input, and a bilinear pairing $\hat{e'} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$ output $\hat{e'}(P, H)^{\alpha^{l+1}} \in \mathbb{G}_T$. Since $\alpha^{l+1} P$ is not an input, the bilinear pairing is of no real use in this regard. Using the shorthand $P_i = \alpha^i P$, an algorithm $\mathcal{A}$ is said to have an advantage ϵ in solving l-BDHE if $Pr[\mathcal{A}(H, P, P_1, P_2, \cdots, P_l, P_{l+2} \cdots, P_{2l}) = \hat{e'}(P_{l+1}, H)] \geq \epsilon$, where the probability is over the random choice of $H, P \in \mathbb{G}$, random choice of $\alpha \in \mathbb{Z}_q$ and random bits used by $\mathcal{A}$. The decisional version of l-BDHE for elliptic curve subgroups may be analogously defined. Let $Y_{(P,\alpha,l)} = (P_1, P_2, \cdots, P_l, P_{l+2} \cdots, P_{2l})$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving decisional l-BDHE in $\mathbb{G}$ if $|Pr[\mathcal{B}(P, H, Y_{(P,\alpha,l)}, \hat{e'}(P_{l+1}, H)) = 0]$-$Pr[\mathcal{B}(P, H, Y_{(P,\alpha,l)}, T) = 0]| \geq \epsilon$, where the probability is over the random choice of $H, P \in \mathbb{G}$, random choice of $\alpha \in \mathbb{Z}_q$, random choice of $T \in \mathbb{G}_T$ and random bits used by $\mathcal{B}$. We refer to the left and right probability distributions as L-BDHE and R-BDHE respectively. Thus, it can be said that the decision (τ, ϵ, l)-BDHE assumption for elliptic curves holds in $\mathbb{G}$ if no τ-time algorithm has advantage ϵ in solving the decisional l-BDHE problem over elliptic curve subgroup $\mathbb{G}$.

The Second Complexity Assumption: We next define the (l, l)-BDHE problem over a pair of equi-prime order bilinear elliptic curve subgroups $\mathbb{G}_1$ with generator P and $\mathbb{G}_2$ with generator Q. Given a vector of $3l + 2$ elements $(H, P, Q, \alpha P, \alpha^2 P, \cdots, \alpha^l P, \alpha^{l+2} P \cdots, \alpha^{2l} P, \alpha Q, \alpha^2 Q, \cdots, \alpha^l Q)$ as input, where P and $\alpha^i P \in \mathbb{G}_1$ and $H, Q, \alpha^i Q \in \mathbb{G}_2$, along with a bilinear pairing $\hat{e''} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, output $\hat{e'}(P, H)^{\alpha^{l+1}} \in \mathbb{G}_T$. Since $\alpha^{l+1} P$ is not an input, the bilinear pairing is of no real use in this regard. Using the shorthand $P_i = \alpha^i P$ and $Q_i = \alpha^i Q$, an algorithm $\mathcal{A}$ is said to have an advantage ϵ in solving (l, l)-BDHE if $Pr[\mathcal{A}(H, P, Q, P_1, P_2, \cdots, P_l, P_{l+2} \cdots, P_{2l}, Q_1, \cdots, Q_l) = \hat{e'}(P_{l+1}, H)] \geq \epsilon$ where the probability is over the random choice of $P \in \mathbb{G}_1$, $H, Q \in \mathbb{G}_2$, random choice of $\alpha \in \mathbb{Z}_q$ and random bits used by $\mathcal{A}$. We may also define the decisional (l, l)-BDHE problem over elliptic curve subgroup pairs as follows. Let $Y_{(P,\alpha,l)} = (P_1, P_2, \cdots, P_l, P_{l+2} \cdots, P_{2l})$ and $Y'_{(Q,\alpha,l)} = (Q_1, Q_2, \cdots, Q_l)$. Also let H be a random element in $\mathbb{G}_2$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving decisional (l, l)-BDHE if $|Pr[\mathcal{B}(P, Q, H, Y_{(P,\alpha,l)}, Y'_{(Q,\alpha,l)}, \hat{e'}(P_{l+1}, H)) = 0]$-$Pr[\mathcal{B}(P, Q, H, Y_{(P,\alpha,l)}, Y'_{(Q,\alpha,l)}, T) = 0]| \geq \epsilon$, where the probability is over the random choice of $P \in \mathbb{G}_1$, $H, Q \in \mathbb{G}_2$, random choice of $\alpha \in \mathbb{Z}_q$, random choice of $T \in \mathbb{G}_T$ and random bits used by $\mathcal{B}$. We refer to the left and right probability distributions as L'-BDHE and R'-BDHE respectively. Thus, it can be said that the decision (τ, ϵ, l, l)-BDHE assumption for elliptic curves holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no τ-time algorithm has advantage ϵ in solving the decisional (l, l)-BDHE problem over elliptic curve subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$. To the best of our knowledge, the (l, l)-BDHE problem has not been introduced in literature before.

Proving the Validity of the Second Complexity Assumption: We prove here that the decision (τ, ϵ, l, l)-BDHE assumption for elliptic curves holds in equi-prime order subgroups $(\mathbb{G}_1, \mathbb{G}_2)$ if the decision (τ, ϵ, l)-BDHE assumption for elliptic curves holds in $\mathbb{G}_1$. Let $\hat{e'} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$ and $\hat{e''} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be bilinear pairings. Also, let P and Q are the generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We first make the following observation.

Observation 1: Since G_1 and G_2 have the same prime order (say q), there exists a bijection $\varphi : \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ such that $\varphi(aP) = aQ$ for all $a \in \mathbb{Z}_q$. Similarly, since $\mathbb{G}_T$ also has order q, there also exists a mapping $\phi : \mathbb{G}_T \longrightarrow \mathbb{G}_T$ such that $\phi(\hat{e'}(H_1, H_2)) = \hat{e''}(H_1, \varphi(H_2))$ for all $H_1, H_2 \in \mathbb{G}_1$.

Let $\mathcal{A}$ be a τ-time adversary that has advantage greater than ϵ in solving the decision (l, l)-BDHE problem over equi-prime order subgroups $(\mathbb{G}_1, \mathbb{G}_2)$. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the l-BDHE problem in $\mathbb{G}_1$. Algorithm $\mathcal{B}$ takes as input a random l-BDHE challenge $(P, H, Y_{(P,\alpha,l)}, Z)$ where Z is either $\hat{e'}(P_{l+1}, H)$ or a random value in $\mathbb{G}_T$. $\mathcal{B}$ computes $Y'_{Q,\alpha,l}$ by setting $Q_i = \varphi(P_i)$ for $i = 1, 2, \cdots, l$. $\mathcal{B}$ also computes $H' = \varphi(H) \in \mathbb{G}_2$ and $Z' = \phi(Z) \in \mathbb{Z}$. Then randomly chooses a bit $b \in (0, 1)$ and sets T_b as Z' and T_{1-b} as a random element in $\mathbb{G}_T$. The challenge given to $\mathcal{A}$ is $((P, Q, H', Y_{(P,\alpha,l)}, Y'_{Q,\alpha,l}), T_0, T_1)$. Quite evidently, when $Z = \hat{e'}(P_{l+1}, H)$ (i.e. the input to $\mathcal{B}$ is a l-BDHE tuple), then $((P, Q, H', Y_{(P,\alpha,l)}, Y'_{Q,\alpha,l}), T_0, T_1)$ is

a valid challenge to A. This is because in such a case, $T_b = Z' = \phi(Z) = \phi(\hat{e'}(P_{l+1}, H)) = \hat{e''}(P_{l+1}, H')$. On the other hand, if Z is a random element in $\mathbb{G}_T$ (i.e. the input to $\mathcal{B}$ is a random tuple), then T_0 and T_1 are just random independent elements of $\mathbb{G}_T$.

Now, $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 0 (indicating that $Z = \hat{e'}(P_{l+1}, H)$). Otherwise, it outputs 1 (indicating that Z is random in $\mathbb{G}_T$). A simple analysis reveals that if $(P, H, Y_{(P,\alpha,l)}, Z)$ is sampled from R-BDHE, $Pr[\mathcal{B}(G, H, Y_{(P,\alpha,l)}, Z) = 0] = \frac{1}{2}$, while if $(P, H, Y_{(P,\alpha,l)}, Z)$ is sampled from L-BDHE, $|Pr[\mathcal{B}(G, H, Y_{(P,\alpha,l)}, Z)] - \frac{1}{2}| \geq \epsilon$. So, the probability that $\mathcal{B}$ outputs correctly is at least ϵ, which in turn implies that $\mathcal{B}$ has advantage at least ϵ in solving the l-BDHE problem. This concludes the proof.

4 The Proposed Dynamic Key-Aggregate Cryptosystem: The Basic Case

In this section, we present the design of our proposed dynamic key-aggregate storage scheme on additive elliptic curve subgroups assuming that there are n ciphertext classes. Our scheme ensures that the ciphertext and aggregate key are of constant size, while the public parameter size is linear in the number of ciphertext classes. Unlike the scheme proposed in [3], the proposed scheme allows dynamic revocation of user access rights without having to massively change the system parameters. We also present a proof of security for the proposed scheme.

4.1 The Basic Construction of Dynamic KAC

Let $\mathbb{G}$ be an additive cyclic elliptic curve subgroup of prime order q, where $2^\lambda \leq q \leq 2^{\lambda+1}$, such that the point P is a generator for $\mathbb{G}$. Also, let $\mathbb{G}_T$ be a multiplicative group of order q with identity element 1. We assume that there exists an efficiently computable bilinear pairing $\hat{e'} : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. We now present the basic construction of our proposed key-aggregate encryption scheme.

The scheme consists of the following five phases.

1. **Setup**$(1^\lambda, n)$: Randomly pick $\alpha \in \mathbb{Z}_q$. Compute $P_i = \alpha^i P \in \mathbb{G}$ for $i = 1, \cdots, n, n+2, \cdots, 2n$. Output the system parameter as $param = (P, P_1, \cdots, P_n, P_{n+2}, \cdots, P_{2n})$. The system also randomly chooses a secret parameter $t \in \mathbb{Z}_q$ which is not made public. It is only known to data owners with credentials to control client access rights.
2. **Keygen**(): Pick $\gamma \in \mathbb{Z}_q$, output the public and master-secret key pair: $(PK = \gamma P, msk = \gamma)$.
3. **Encrypt**(PK, i, m): For a message $m \in \mathbb{G}_T$ and an index $i \in \{1, 2, \cdots, n\}$, randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Then the ciphertext is computed as $\mathcal{C} = (rP, t'(PK + P_i), m.\hat{e'}(P_n, t'P_1)) = (c_1, c_2, c_3)$

4. **Extract**$(msk = \gamma, \mathcal{S})$: For the set $\mathcal{S}$ of indices j the aggregate key is computed as
$K_{\mathcal{S}} = \sum_{j \in \mathcal{S}} \gamma P_{n+1-j} = \sum_{j \in \mathcal{S}} \alpha^{n+1-j} PK$
and the dynamic access control parameter U is computed as tP. Thus the net aggregate key is $(K_{\mathcal{S}}, U)$ which is transmitted via a secure channel to users that have access rights to $\mathbb{S}$.
5. **Decrypt**$(K_{\mathcal{S}}, U, \mathcal{S}, i, \mathcal{C} = \{c_1, c_2, c_3\})$: If $i \notin \mathcal{S}$, output $\perp$. Otherwise return the message $\hat{m} = c_3 \hat{e'}(K_{\mathcal{S}} + \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i}, U + c_1) / (\hat{e'}(\sum_{j \in \mathcal{S}} P_{n+1-j}, c_2))$.

The proof of correctness of this scheme is presented below.

$$
\begin{aligned}
\hat{m} &= c_3 \frac{\hat{e'}(K_{\mathcal{S}} + \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i}, U + c_1)}{\hat{e'}(\sum_{j \in \mathcal{S}} P_{n+1-j}, c_2)} \\
&= c_3 \frac{\hat{e'}(\sum_{j \in \mathcal{S}} \gamma P_{n+1-j}, t'P) \hat{e'}(\sum_{j \in \mathcal{S}} (P_{n+1-j+i}) - P_{n+1}, t'P)}{\hat{e'}(\sum_{j \in \mathcal{S}} P_{n+1-j}, t'PK) \hat{e'}(\sum_{j \in \mathcal{S}} P_{n+1-j}, t'P_i))} \\
&= c_3 \frac{\hat{e'}(\sum_{j \in \mathcal{S}} P_{n+1-j+i}, t'P)}{\hat{e'}(P_{n+1}, t'P) \hat{e'}(\sum_{j \in \mathcal{S}} P_{n+1-j+i}, t'P))} \\
&= m
\end{aligned}
$$

4.2 Dynamic Access Control

An important aspect of the proposed scheme is the fact that it allows the data owner to dynamically update user access permissions. In KAC [3], once the data owner issues an aggregate key corresponding to a set of ciphertext classes to a user, revoking the user's access permissions to the same is not possible without changing the master secret key. However, changing the master secret key each time an user's access privileges to a ciphertext class need to be updated, is a very expensive option and may not be practically feasible. Our scheme, on the other hand, offers a solution to this problem by allowing the data owner to dynamically update user access privileges.

We achieve this by making the parameter $U = tP$ a part of the aggregate key in our proposed scheme and not a part of the ciphertext. The user must have the correct value of U in possession to be able to decrypt any encrypted ciphertext class in the subset $\mathcal{S}$. Now suppose the data owner wishes to alter the access rights to the subset $\mathcal{S}$. She can simply re-encrypt all ciphertexts in that class using a different random element $\hat{t} \in \mathbb{Z}_q$, and then provide the updated dynamic access parameter $\hat{U} = \hat{t}P$ to only those users who she wishes to delegate access to. The decrypted value will give the correct message m only if the same t is used for both encryption and decryption. This is a major difference between our scheme and the scheme proposed in [3], where the knowledge of the random parameter was only embedded as part of the ciphertext itself, and could not be used to control access rights of users. Moreover, since U is of constant size and needs to be transmitted only when changed (and not for every encryption), there is no significant degradation in performance.

4.3 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. A suggested in [4], 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.

It is important to note that our proposed scheme fixes the number of ciphertext classes beforehand, thus limiting the scope for ciphertext class extension. The only way to increase the number of classes is to change the public key parameters, which would therefore require some kind of administrative privileges, and cannot be done by an user for her own purposes. However, in online data sharing environments, users may wish to register their own public key-private key pairs for new ciphertext classes according to their own requirements. Such an extension to the scheme would make extremely convenient and attractive to potential users. A proposal made in [3] recommends that the user be allowed to register new public-private key pairs, at the cost of increasing the number of ciphertext classes by n each time. This is both impractical and wasteful. In the next section, we present a two-tier generalization of our scheme that tackles this issue in a more economical fashion. We avoid a separate proof of semantic security for the base case presented here, since the proof is a special case of the proof for the generalized scheme presented in the next section.

5 A Generalized Version of Dynamic KAC

In this section, we focus on building an efficiently extensible version of our proposed scheme that allows an user to economically increase the number of ciphertext classes while registering a new public key-private key pair. We adopt the idea presented in [4] to develop a hierarchical structure that has multiple instances (say n_1) of the original scheme running in parallel. Each such instance in turn provides *locally aggregate keys* for n_2 ciphertext sub-classes. Each ciphertext class thus now has a double index (i_1, i_2) where $1 \leq i_1 \leq n_1$ and $1 \leq i_2 \leq n_2$. This allows the overall setup to handle $n = n_1 n_2$ classes. However, it is important to note that all the instances can use the same public parameters. This interaction among the instances helps to largely improve performance. We further point out that while in [4], the generalized construction offers a trade-off between the public parameter size and the ciphertext size, our generalized scheme actually reduces the public parameter size without compromising on the size of the ciphertext. Further, addition of a single new key increases the number of classes only by n_2 and not by n. Setting $n_2 \ll n$ thus achieves significant improvement in performance over the existing proposal.

5.1 The Construction of the Generalized KAC

Let n_2 be a fixed positive integer. Our proposed n_2-generalized key-aggregate encryption scheme over elliptic curve subgroups is as described below. It may be noted that the bilinear additive elliptic curve sub-group $\mathbb{G}$ and the multiplicative group $\mathbb{G}_T$, as well as the pairing $\hat{e'}$ are the same as in the basic scheme. The algorithm sets up $n_1 = \lfloor n/n_2 \rfloor$ instances of the basic scheme, each of which handles n_2 ciphertext classes. The original scheme is thus a special case of the extended scheme with $n_1 = 1$ and $n_2 = n$.

1. **Setup**$(1^\lambda, n_2)$: Randomly pick $\alpha \in \mathbb{Z}_q$. Compute $P_i = \alpha^i P \in \mathbb{G}$ for $i = 1, \cdots, n_2, n_2 + 2, \cdots, 2n_2$. Output the system parameter as $param = (P, P_1, \cdots, P_{n_2}, P_{n_2+2}, \cdots, P_{2n_2})$. The system randomly chooses a secret parameter $t \in \mathbb{Z}_q$ which is not made public. It is only known to data owners with credentials to control client access rights.
2. **Keygen**(): Pick $\gamma_1, \gamma_2, \cdots, \gamma_{n_1} \in \mathbb{Z}_q$, output the public and master-secret key pair:
$(PK{=}(pk_1, pk_2, \cdots, pk_{n_1}) = (\gamma_1 P, \gamma_2 P, \cdots, \gamma_{n_1} P), msk{=}(\gamma_1, \gamma_2, \cdots, \gamma_{n_1}))$.
3. **Encrypt**$(pk_{i_1}, (i_1, i_2), m)$: For a message $m \in \mathbb{G}_T$ and an index $(i_1, i_2) \in \{1, 2, \cdots, n_1\} \times \{1, 2, \cdots, n_2\}$, randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Then compute the ciphertext $\mathcal{C}{=}(rP, t'(pk_{i_1} + P_{i_2}), m.\hat{e'}(P_{n_2}, t'P_1)) = (c_1, c_2, c_3)$.
4. **Extract**$(msk = \gamma, \mathcal{S})$: For the set $\mathcal{S}$ of indices (j_1, j_2) the aggregate key is computed as $K_\mathcal{S} = (k_\mathcal{S}^1, k_\mathcal{S}^2, \cdots, k_\mathcal{S}^{n_1}) =$
$(\sum_{(1,j_2)\in\mathcal{S}} \gamma_1 P_{n_2+1-j_2}, \sum_{(2,j_2)\in\mathcal{S}} \gamma_2 P_{n_2+1-j_2}, \cdots, \sum_{(n_1,j_2)\in\mathcal{S}} \gamma_{n_1} P_{n_2+1-j_2})$
and the dynamic access control parameter U is computed as tP. Thus the net aggregate key is $(K_\mathcal{S}, U)$ which is transmitted via a secure channel to users that have access rights to $\mathcal{S}$. Note that $k_\mathcal{S}^{j_1} = \sum_{(j_1,j_2)\in\mathcal{S}} \alpha^{n+1-j_2} pk_{j_1}$ for $j_1 = 1, 2, \cdots, n_1$.
5. **Decrypt**$(K_\mathcal{S}, U, \mathcal{S}, (i_1, i_2), \mathcal{C} = \{c_1, c_2, c_3\})$: If $(i_1, i_2) \notin \mathcal{S}$, output $\perp$. Otherwise return the message
$\hat{m} = c_3 \frac{\hat{e'}(k_\mathcal{S}^{i_1} + \sum_{(i_1,j_2)\in\mathcal{S}, j_2\neq i_2} P_{n_2+1-j_2+i_2}, U+c_1)}{\hat{e'}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2}, c_2)}$.

The proof of correctness for the generalized scheme is very similar to that for the basic scheme.

5.2 Semantic Security of the Generalized KAC

The Reduced Generalized Scheme: We define a reduced version of the generalized encryption scheme. We note that the ciphertext $\mathcal{C} = (c_1, c_2, c_3)$ output by the *Encypt* operation essentially embeds the value of m in c_3 by multiplying it with $\hat{e'}(P_{n_2}, tP_1)$. Consequently, the security of our proposed scheme is equivalent to that of a *reduced* generalized key-aggregate encryption scheme that simply uses the reduced ciphertext (c_1, c_2), the aggregate key $K_\mathcal{S}$ and the dynamic access parameter U to successfully transmit and decrypt the value of $\hat{e'}(P_{n_2}, t'P_1) = \hat{e'}(P_{n_2+1}, t'P)$. We prove the semantic security of this *reduced*

scheme parameterized with a given number of ciphertext classes n_2 for each instance, which also amounts to proving the semantic security of our original encryption scheme for the same number of ciphertext classes. Note that the proof of security is independent of the number of instances n_1 that run in parallel.

The Adversarial Model: We make the following assumptions about the adversary $\mathcal{A}$:

1. The adversary has the aggregate key that allows her to access any ciphertext class other than those in the target subset $\mathcal{S}$, that is, she possesses $K_{\overline{\mathcal{S}}}$.
2. The adversary has access to the public parameters *param* and PK, and also possesses the dynamic access parameter U.

The Security Proof: The security proof presented here uses the first complexity assumption stated in Sect. 3.3 (The First Complexity Assumption). Let $\mathbb{G}$ be a bilinear elliptic curve subgroup of prime order q and G_T be a multiplicative group of order q. Let $\hat{e'} : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be a bilinear non-degenerate pairing. For any pair of positive integers $n_2, n'(n' > n_2)$ our proposed n_2-generalized reduced key-aggregate encryption scheme over elliptic curve subgroups is (τ, ϵ, n') semantically secure if the decision (τ, ϵ, n_2)-BDHE assumption holds in $\mathbb{G}$. We now prove this statement below.

Proof: Let for a given input n', $\mathcal{A}$ be a τ-time adversary that has advantage greater than ϵ for the *reduced scheme* parameterized with a given n_2. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the n_2-BDHE problem in $\mathbb{G}$. Algorithm $\mathcal{B}$ takes as input a random n_2-BDHE challenge $(P, H, Y_{(P,\alpha,n_2)}, Z)$ where Z is either $\hat{e'}(P_{n_2+1}, H)$ or a random value in $\mathbb{G}_T$. Algorithm $\mathcal{B}$ proceeds as follows.

1. **Init:** Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}$ of ciphertext classes that $\mathcal{A}$ wishes to be challenged on. For each ciphertext class $(i_1, i_2) \in \mathcal{S}$, $\mathcal{B}$ performs the **SetUp**-$(\mathbf{i_1}, \mathbf{i_2})$, **Challenge**-$(\mathbf{i_1}, \mathbf{i_2})$ and **Guess**-$(\mathbf{i_1}, \mathbf{i_2})$ steps. Note that the number of iterations is polynomial in $|S|$.
2. **SetUp**-$(\mathbf{i_1}, \mathbf{i_2})$: $\mathcal{B}$ should generate the public *param*, public key PK, the access parameter U, and the aggregate key $K_{\overline{\mathcal{S}}}$. For the iteration corresponding to ciphertext class (i_1, i_2), they are generated as follows.
   - *param* is set as (P, Y_{P,α,n_2}).
   - Randomly generate $u_1, u_2, \cdots, u_{n_1} \in \mathbb{Z}_q$. Then, set $PK = (pk_1, pk_2, \cdots, pk_{n_1})$, with $pk_{j_1} = u_{j_1} P - P_{i_2}$ for $j_1 = 1, 2, \cdots, n_1$.
   - Set $K_{\overline{\mathcal{S}}} = (k^1_{\overline{\mathcal{S}}}, k^2_{\overline{\mathcal{S}}}, \cdots, k^{n_1}_{\overline{\mathcal{S}}})$, where $k^{j_1}_{\overline{\mathcal{S}}}$ is set as $\sum_{(j_1,j_2)\notin\mathcal{S}} (u_{j_1} P_{n_2+1-j_2} - (P_{n_2+1-j_2+i_2}))$. Then, $k^{j_1}_{\overline{\mathcal{S}}} = \sum_{(j_1,j_2)\notin\mathcal{S}} \alpha^{n_2+1-j_2} pk_{j_1}$,which is as per the scheme specification. Note that $\mathcal{B}$ knows that $(i_1, i_2) \notin \overline{\mathcal{S}}$, and hence has all the resources to compute this aggregate key for $\overline{\mathcal{S}}$.
   - U is set as some random element in $\mathbb{G}$.

   Note that since P, α, U and the u_{j_1} values are chosen uniformly at random, the public key has an identical distribution to that in the actual construction.

3. **Challenge-($\mathbf{i_1}, \mathbf{i_2}$)**: To generate the challenge for the ciphertext class (i_1, i_2), $\mathcal{B}$ computes (c_1, c_2) as $(H - U, u_{i_1} H)$. It then randomly chooses a bit $b \in (0, 1)$ and sets K_b as Z and K_{1-b} as a random element in $\mathbb{G}_T$. The challenge given to $\mathcal{A}$ is $((c_1, c_2), K_0, K_1)$.
   We claim that when $Z = \hat{e}'(P_{n_2+1}, H)$ (i.e. the input to $\mathcal{B}$ is a n_2-BDHE tuple), then $((c_1, c_2), K_0, K_1)$ is a valid challenge to A. We prove this claim here. we point out that P is a generator of $\mathbb{G}$ and so $H = t'P$ for some $t' \in \mathbb{Z}_q$. Putting H as $t'P$ gives us the following:
   - $U = tP$ for some $t \in \mathbb{Z}_q$
   - $c_1 = H - U = (t' - t)P = rP$ for $r = t' - t$
   - $c_2 = u_{i_1} H = (u_{i_1}) t' P = t'(u_{i_1} P) = t'(u_{i_1} P - P_{i_2} + P_{i_2}) = t'(pk_{i_1} + P_{i_2})$
   - $K_b = Z = \hat{e}'(P_{n_2+1}, H) = \hat{e}'(P_{n_2+1}, t'P)$

   On the other hand, if Z is a random element in $\mathbb{G}_T$ (i.e. the input to $\mathcal{B}$ is a random tuple), then K_0 and K_1 are just random independent elements of $\mathbb{G}_T$.
4. **Guess-($\mathbf{i_1}, \mathbf{i_2}$)**: The adversary $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 0 (indicating that $Z = \hat{e}'(P_{n_2+1}, H)$), and terminates. Otherwise, it goes for the next ciphertext class in $\mathcal{S}$.

If after $|\mathcal{S}|$ iterations, $b' \neq b$ for each ciphertext class $(i_1, i_2) \in \mathcal{S}$, the algorithm $\mathcal{B}$ outputs 1 (indicating that Z is random in $\mathbb{G}_T$). We now analyze the probability that $\mathcal{B}$ gives a correct output. If $(P, H, Y_{(P,\alpha,n_2)}, Z)$ is sampled from R-BDHE, $Pr[\mathcal{B}(G, H, Y_{(P,\alpha,n_2)}, Z) = 0] = \frac{1}{2}$, while if $(P, H, Y_{(P,\alpha,n_2)}, Z)$ is sampled from L-BDHE, $|Pr[\mathcal{B}(G, H, Y_{(P,\alpha,n_2)}, Z)] - \frac{1}{2}| \geq \epsilon$. So, the probability that $\mathcal{B}$ outputs correctly is at least $1 - (\frac{1}{2} - \epsilon)^{|\mathcal{S}|} \geq \frac{1}{2} + \epsilon$. Thus $\mathcal{B}$ has advantage at least ϵ in solving the n_2-BDHE problem. This concludes the proof. *Note that the instance of this proof with $n_1 = 1$ and $n_2 = n$ serves as the proof of security for the basic KAC scheme proposed in Sect. 4.*

Performance Trade Off with the Basic Scheme: We compare the various parameter sizes for the proposed original and extended schemes in Table 1. We note that *SetUp* and *KeyGen* are both one-time operations, and for a given subset $\mathcal{S}$, the *Extract* operation is also performed once to generate the corresponding aggregate key $K_{\mathcal{S}}$. The most important advantage that the generalized scheme provides is the user's ability to efficiently extend the number of ciphertext classes. As far as encryption and decryption are concerned, encryption should ideally take the same time for both schemes, while decryption is actually expected to be faster for the generalized construction as $n_2 \leq n$.

5.3 A Flexible Extension Policy

If a user needs to classify her ciphertexts into more that n classes, she can register for additional key pairs $(pk_{n_1+1}, msk_{n_1+1})$, $\cdots$, $(pk_{n_1+l}, msk_{n_1+l})$ as per her requirements. Each new key registration increases the number of classes by n_2,

Table 1. Comparison between the basic and generalized schemes

Item	Nature of computation	Original scheme	Generalized scheme
$param$(SetUp)	One-time	$\mathcal{O}(n)$	$\mathcal{O}(n_2)$
PK(KeyGen)	One-time	$\mathcal{O}(1)$	$\mathcal{O}(n_1)$
$K_{\mathcal{S}}$(Extract)	One-time	$\mathcal{O}(1)$	$\mathcal{O}(n_1)$
$\mathcal{C}$	One per message	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Encrypt	One per message	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Decrypt	One per message	$\mathcal{O}(\|\mathcal{S}\|)$	$\mathcal{O}(\|\mathcal{S}\|)$

where $n_2 \leq n$. The idea of under-utilization stems from the fact that registration of each public-private key pair increases the number of classes by n_2. However, it is not necessary that all the existing classes are utilized at any given point of time. For instance, a user may at any point of time want to register l new private-public key pairs, however she will in all probability not use up all ln_2 additional classes of messages that could be encrypted using the newly registered keys. We stress here is that, unlike in the public key extension scheme proposed in [3] where the values of n_1 and n_2 are fixed to 1 and n respectively, our generalized construction *provides a choice* of n_1 and n_2 so that the system administrator could choose pair of values suited to their requirements.

We propose a metric to quantify the under-utilization of ciphertext classes for a given configuration of the system. Let us assume that at some instance of time, there are $n_1 + l$ private-public key pairs registered in the system, and c_i classes corresponding to each key are being utilized. We define the utilization coefficient as $\frac{1}{1+\xi}$, where $\xi = -\frac{1}{n_1}\sum_{i=1 c_i \neq 0}^{n_1} \log(\frac{c_i}{n_2})$. An efficient scheme tries to minimize the value of ξ to achieve good utilization of the existing set of classes. The value is maximum when $c_i = n_2 \forall i = 1, 2, \cdots, n_2$. Note that $c_i = 0$ implies that no subclasses under the given key pk_i are being utilized, which is equivalent to not registering the key at all.

To stress the importance of the flexible extension policy, we provide a simplified example here. We consider two possible configurations of the extended scheme. In the first configuration, $n_1 = 1$ and $n_2 = n$, which is essentially identical to the public key extension scheme proposed in [3]. The other configuration has $n_1 > 1$ and $n_2 < n$. Now assume that before extension, both schemes utilized c ciphertext classes out of the n possible classes, equally distributed across all key pairs. Now suppose a situation arises where an user needs to register l more key pairs, and utilizes $z < n_2$ classes corresponding to each key. In the first configuration, we have $\xi_1 = -\frac{1}{l+1}(l\log(\frac{z}{n}) + \log(\frac{c}{n}))$, while for the second configuration, $\xi_2 = -\frac{1}{l+n_1}(l(\log(\frac{z}{n_2})) + n_1\log(\frac{c}{n}))$. Now for $l > (\frac{n_1}{\log n_1} - 1)\log(\frac{z}{c}) - 1$, $\xi_2 < \xi_1$. Thus for any value of (n_1, n_2) other than $(1, n)$, there exists a value of l for which the scheme achieves better utilization coefficient. Since l is expected to increase in a dynamic scenario, our public key extension scheme eventually performs better than the scheme suggested in [3].

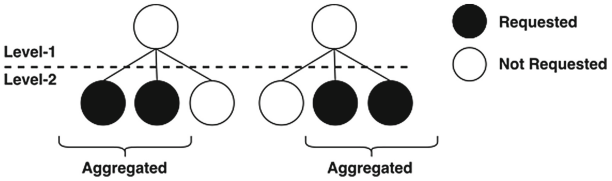

Fig. 2. A practical request scenario in the hierarchical setting

5.4 Advantage over Hierarchical Encryption Based Schemes

Although the generalized scheme has a two level hierarchy (with each of the n_1 parallely executing instances of the basic scheme representing a node in the top level and the actual ciphertext classes representing nodes in the lower level), it avoids the pitfalls of existing hierarchical encryption based schemes [5,10]. In standard tree based hierarchical systems, granting access to the key corresponding to any node implicitly grants access to all the keys in the subtree rooted at that node. This means granting access to a selected set of nodes in a given subtree would blow up the key-size to be the same as the number of nodes. This is avoided in our generalized scheme, since any number of nodes (ciphertext classes) that belong to the same instance may be aggregated into a single key. Figure 2 summarizes this phenomenon. In the situation depicted, a tree-based hierarchy system would require 4 decryption keys, while our scheme would require only 2. In this respect, our scheme has similar advantages to that of [3].

6 Extending the Generalized KAC for Efficient Pairings on Elliptic Curve Subgroups

The encryption schemes proposed so far use the assumption that the elliptic curve pairing bilinear pairing $\hat{e'} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$ satisfies the property $\hat{e'}(P, P) \neq 1$, where P is the generator for G_1. In this section, we propose an extension to the generalized n_2-scheme that allows using pairings of the form $\hat{e''} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, where G_1 and G_2 are two elliptic curve subgroups of the same prime order. The motivation behind this extension is that many popular pairing algorithms such as the Tate [11], Eta [12], and Ate [13] pairings are defined over two distinct elliptic curve subgroups G_1 and G_2 of the same order. Many efficient implementations of such pairings on sensor nodes such as TinyTate [14] have been proposed in literature. This motivates us to modify our scheme in a manner that allows using such well-known pairings. The modified encryption scheme described below allows using a pairing $\hat{e''} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with P generator of G_1 and Q generator of G_2.

6.1 Construction of the Extended KAC

1. **Setup**$(1^\lambda, n_2)$: Randomly pick $\alpha \in \mathbb{Z}_q$. Compute $P_i = \alpha^i P \in \mathbb{G}_1$ for $i = 1, \cdots, n_2, n_2 + 2, \cdots, 2n_2$ and $Q_i = \alpha^i Q \in \mathbb{G}_2$ for $i = 1, \cdots, n_2$.

Output the system parameter as $param = (P, P_1, \cdots, P_{n_2}, P_{n_2+2}, \cdots, P_{2n_2}, Q, Q_1, \cdots, Q_{n_2})$. The system also randomly chooses secret parameters $t \in \mathbb{Z}_q$ which is not made public. It is only transferred through a secure channel to data owners with credentials to control client access rights.

2. **Keygen**(): Pick $\gamma_1, \gamma_2, \cdots, \gamma_{n_1} \in \mathbb{Z}_q$, output the public and master-secret key tuple:
$(PK^1 = (pk^1{}_1, pk^1{}_2, \cdots, pk^1{}_{n_1}) = (\gamma_1 P, \gamma_2 P, \cdots, \gamma_{n_1} P)$, $PK^2 =$
$(pk^2{}_1, pk^2{}_2, \cdots, pk^2{}_{n_1}) = (\gamma_1 Q, \gamma_2 Q, \cdots, \gamma_{n_1} Q)$, $msk = (\gamma_1, \gamma_2, \cdots, \gamma_{n_1}))$.
3. **Encrypt**$(pk_{i_1}, (i_1, i_2), m)$: For a message $m \in \mathbb{G}_T$ and an index $(i_1, i_2) \in \{1, 2, \cdots, n_1\} \times \{1, 2, \cdots, n_2\}$, randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Then compute the ciphertext as
$\mathcal{C} = (rQ, t'(pk^2{}_{i_1} + Q_{i_2}), m.\hat{e''}(P_{n_2}, t'Q_1)) = (c_1, c_2, c_3)$.
4. **Extract**$(msk = \gamma, \mathcal{S})$: For the set $\mathcal{S}$ of indices (j_1, j_2) the aggregate key is computed as $K_{\mathcal{S}} = (k^1_{\mathcal{S}}, k^2_{\mathcal{S}}, \cdots, k^{n_1}_{\mathcal{S}}) =$
$(\sum_{(1,j_2)\in\mathcal{S}} \gamma_1 P_{n_2+1-j_2}, \sum_{(2,j_2)\in\mathcal{S}} \gamma_2 P_{n_2+1-j_2}, \cdots, \sum_{(n_1,j_2)\in\mathcal{S}} \gamma_{n_1} P_{n_2+1-j_2})$
and the dynamic access control parameter U is computed as tQ. Thus the net aggregate key is $(K_{\mathcal{S}}, U)$ which is transmitted via a secure channel to users that have access rights to $\mathcal{S}$. Note that $k^{j_1}_{\mathcal{S}} = \sum_{(j_1,j_2)\in\mathcal{S}} \alpha^{n+1-j_2} pk^1{}_{j_1}$ for $j_1 = 1, 2, \cdots, n_1$.
5. **Decrypt**$(K_{\mathcal{S}}, U, \mathcal{S}, (i_1, i_2), \mathcal{C} = \{c_1, c_2, c_3\})$: If $(i_1, i_2) \notin \mathcal{S}$, output $\perp$. Otherwise return the message
$\hat{m} = c_3 \frac{\hat{e''}(k^{i_1}_{\mathcal{S}} + \sum_{(i_1,j_2)\in\mathcal{S}, j_2\neq i_2} P_{n_2+1-j_2+i_2}, U+c_1)}{\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2}, c_2)}$.

The proof of correctness of this scheme is presented below.

$$
\begin{aligned}
\hat{m} &= c_3 \frac{\hat{e''}(k^{i_1}_{\mathcal{S}} + \sum_{(i_1,j_2)\in\mathcal{S}, j_2\neq i_2} P_{n_2+1-j_2+i_2}, U + c_1)}{\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2}, c_2)} \\
&= c_3 \frac{\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} \gamma_{i_1} P_{n_2+1-j_2}, t'Q)\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} (P_{n_2+1-j_2+i_2}) - P_{n_2+1}, t'Q)}{\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2}, \gamma_{i_1}(t'Q))\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2}, \alpha^{i_2}(t'Q))} \\
&= c_3 \frac{\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2+i_2}, t'Q)}{\hat{e''}(P_{n_2+1}, t'Q)\hat{e''}(\sum_{(i_1,j_2)\in\mathcal{S}} P_{n_2+1-j_2+i_2}, t'Q)} \\
&= m
\end{aligned}
$$

6.2 Semantic Security of the Extended KAC

The proof of security uses a reduced version of the extended KAC scheme, analogous to the reduced scheme used for proving the security of the generalized KAC. The adversarial model is also the assumed to be the same as for the generalized KAC. The proof uses the (l, l)-BDHE assumption proposed in Sect. 3.3 (The Second Complexity Assumption). Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additive elliptic curve subgroups of prime order q, and G_T be a multiplicative group of order q. Let $\hat{e''} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a bilinear non-degenerate pairing. We claim that for any pair of positive integers $n_2, n'(n' > n_2)$ our proposed extension to the

n_2-generalized reduced key-aggregate encryption scheme over elliptic curve subgroups is (τ, ϵ, n') semantically secure if the decision $(\tau, \epsilon, n_2, n_2)$-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. We prove the claim below.

Proof: Let for a given input n', $\mathcal{A}$ be a τ-time adversary that has advantage greater than ϵ for the *reduced scheme* parameterized with a given n_2. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the (n_2, n_2)-BDHE problem in $\mathbb{G}$. Algorithm $\mathcal{B}$ takes as input a random (n_2, n_2)-BDHE challenge $(P, Q, H, Y_{(P,\alpha,n_2),Y'_{Q,\alpha,n_2}}, Z)$ where Z is either $\hat{e''}(P_{n_2+1}, H)$ or a random value in $\mathbb{G}_T$. Algorithm $\mathcal{B}$ proceeds as follows.

1. **Init:** Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}$ of ciphertext classes that $\mathcal{A}$ wishes to be challenged on. For each ciphertext class $(i_1, i_2) \in \mathcal{S}$, $\mathcal{B}$ performs the **SetUp**-$(\mathbf{i_1}, \mathbf{i_2})$, **Challenge**-$(\mathbf{i_1}, \mathbf{i_2})$ and **Guess**-$(\mathbf{i_1}, \mathbf{i_2})$ steps. Note that the number of iterations is polynomial in $|S|$.
2. **SetUp**-$(\mathbf{i_1}, \mathbf{i_2})$: $\mathcal{B}$ should generate the public *param*, public keys PK^1, PK^2, the access parameter U, and the aggregate key $K_{\overline{\mathcal{S}}}$. For the iteration corresponding to ciphertext class (i_1, i_2), they are generated as follows.
   - *param* is set as $(P, Q, Y_{P,\alpha,n_2}, Y'_{Q,\alpha,n_2})$.
   - Randomly generate $u_1, u_2, \cdots, u_{n_1} \in \mathbb{Z}_q$. Then, set PK^1=$(pk^1{}_1, pk^1{}_2, \cdots, pk^1{}_{n_1})$, where $pk^1{}_{j_1}$ is set as $u_{j_1}P - P_{i_2}$ for $j_1 = 1, 2, \cdots, n_1$, and set PK^2=$(pk^2{}_1, pk^2{}_2, \cdots, pk^2{}_{n_1})$, where $pk^2{}_{j_1}$ is set as $u_{j_1}Q - Q_{i_2}$ for $j_1 = 1, 2, \cdots, n_1$
   - $K_{\overline{\mathcal{S}}}$ is set as $(k^1_{\overline{\mathcal{S}}}, k^2_{\overline{\mathcal{S}}}, \cdots, k^{n_1}_{\overline{\mathcal{S}}})$ where $k^{j_1}_{\overline{\mathcal{S}}}$ $= \sum_{(j_1,j_2)\notin\mathcal{S}}(u_{j_1}P_{n_2+1-j_2} - (P_{n_2+1-j_2+i_2}))$ for $j_1 = 1, 2, \cdots, n_1$. Note that this implies $k^{j_1}_{\overline{\mathcal{S}}} = \sum_{(j_1,j_2)\notin\mathcal{S}} \alpha^{n_2+1-j_2} pk^1{}_{j_1}$, as is supposed to be as per the scheme specification. Note that $\mathcal{B}$ knows that $(i_1, i_2) \notin \overline{\mathcal{S}}$, and hence has all the resources to compute this aggregate key for $\overline{\mathcal{S}}$.
   - U is set as some random element in $\mathbb{G}_2$.

   Note that since P, Q, α, U and the u_{j_1} values are chosen uniformly at random, the public key has an identical distribution to that in the actual construction.
3. **Challenge**-$(\mathbf{i_1}, \mathbf{i_2})$: To generate the challenge for the ciphertext class (i_1, i_2), $\mathcal{B}$ computes (c_1, c_2) as $(H - U, u_{i_1}H)$. It then randomly chooses a bit $b \in (0, 1)$ and sets K_b as Z and K_{1-b} as a random element in $\mathbb{G}_T$. The challenge given to $\mathcal{A}$ is $((c_1, c_2), K_0, K_1)$.
   We claim that when $Z = \hat{e''}(P_{n_2+1}, H)$ (i.e. the input to $\mathcal{B}$ is a n_2-BDHE tuple), then $((c_1, c_2), K_0, K_1)$ is a valid challenge to A. We prove this claim here. we point out that Q is a generator of $\mathbb{G}_2$ and so $H = t'P$ for some $t' \in \mathbb{Z}_q$. Putting H as $t'Q$ gives us the following:
   - $U = tQ$ for some $t \in \mathbb{Z}_q$
   - $c_1 = H - U = (t' - t)Q = rQ$ where $r = t' - t$
   - $c_2 = u_{i_1}H = (u_{i_1})t'Q = t'(u_{i_1}Q) = t'(u_{i_1}Q - Q_{i_2} + Q_{i_2}) = t'(pk^2{}_{i_1} + Q_{i_2})$
   - $K_b = Z = \hat{e''}(P_{n_2+1}, H) = \hat{e''}(P_{n_2+1}, t'Q)$

On the other hand, if Z is a random element in $\mathbb{G}_T$ (i.e. the input to $\mathcal{B}$ is a random tuple), then K_0 and K_1 are just random independent elements of $\mathbb{G}_T$.

4. **Guess-($\mathbf{i_1}, \mathbf{i_2}$)**: The adversary $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 0 (indicating that $Z = \hat{e''}(P_{n+1}, H)$), and terminates. Otherwise, it goes for the next ciphertext class in $\mathcal{S}$.

If after $|\mathcal{S}|$ iterations, $b' \neq b$ for each ciphertext class $(i_1, i_2) \in \mathcal{S}$, the algorithm $\mathcal{B}$ outputs 1 (indicating that Z is random in $\mathbb{G}_T$). We now analyze the probability that $\mathcal{B}$ gives a correct output. If $(P, H, Y_{(P,\alpha,n_2)}, Z)$ is sampled from R'-BDHE, $Pr[\mathcal{B}(G, H, Y_{(P,\alpha,n_2)}, Z) = 0] = \frac{1}{2}$, while if $(P, H, Y_{(P,\alpha,n_2)}, Z)$ is sampled from L'-BDHE, $|Pr[\mathcal{B}(G, H, Y_{(P,\alpha,n_2)}, Z)] - \frac{1}{2}| \geq \epsilon$. So, the probability that $\mathcal{B}$ outputs correctly is at least $1 - (\frac{1}{2} - \epsilon)^{|\mathcal{S}|} \geq \frac{1}{2} + \epsilon$. Thus $\mathcal{B}$ has advantage at least ϵ in solving the (n_2, n_2)-BDHE problem. This concludes the proof.

7 Experimental Results Using Tate Pairings

In this section we present experimental results from our implementations of the extended generalized scheme using Tate pairings on BN-curves using 256 bit primes [15]. All our experiments have been carried out on an AMD Opteron (TM) Processor 6272 × 16 with a clock frequency 1.4 GHz.

7.1 Space and Time Complexities

Table 2 summarizes the space requirements for various parameters of the scheme for different values of (n_1, n_2). The results have been averaged over 100 randomly chosen subsets of the $n = 100$ ciphertext classes. Table 3 summarizes the time complexity for various operations of the scheme for different values of (n_1, n_2). The results have been averaged over 100 randomly chosen subsets of the $n = 100$ ciphertext classes. The encryption and decryption operation complexities are further averaged over 10 message transmissions corresponding to each subset. We point out that both the overall space and time requirements are minimum for $n_1 = n_2 = 10 = \sqrt{n}$, which proves the usefulnesss of the generaalization.

7.2 Comparison with Hierarchy Based Schemes

Next, we compare specifically the key size required for the proposed extended scheme, for different values of n_1 and n_2 (again corresponding to $n = 100$), with that required for a hierarchical encryption construction [16]. Since our scheme uses a hierarchy depth of 2, we use the same for the hierarchical construction as well, with n_1 nodes in level 0, and n_2 level 1 nodes in the subtree rooted at each level 0 node. Figure 3 summarizes the findings. Evidently, lower the value of n_1, better the key aggregation, hence lower the ratio.

Table 2. Space complexities

n_1	n_2	$param$ (in bytes)	PK (in bytes)	msk (in bytes)	K_S (in bytes)	U (in bytes)	Total (in KB)
1	100	16112	144	40	72	64	16.046875
2	50	8112	240	56	120	64	8.390625
4	25	4112	432	88	216	64	4.796875
5	20	3312	528	104	264	64	4.171875
10	**10**	**1712**	**1008**	**184**	**504**	**64**	**3.390625**
20	5	912	1968	344	984	64	4.171875
25	4	752	2448	424	1224	64	4.796875
50	2	432	4848	824	2424	64	8.390625
100	1	272	9648	1624	4824	64	16.046875

Table 3. Time complexities

n_1	n_2	$SetUp$ (in clock cycles)	$KeyGen$ (in clock cycles)	$Encrypt$ (in clock cycles)	$Extract$ (in clock cycles)	$Decrypt$ (in clock cycles)	Total (in clock cycles)
1	100	2920000	10000	7932000	47000	16095000	27004100
2	50	1410000	30000	8065000	53000	16110000	25668000
4	25	690000	60000	8130000	81000	16284000	25245000
5	20	590000	70000	8091000	96000	16379000	25226000
10	**10**	**280000**	**140000**	**7957000**	**170000**	**16049000**	**25136000**
20	5	130000	270000	8070000	320000	16361000	25151000
25	4	120000	350000	8256000	370000	16239000	25836000
50	2	50000	680000	8265000	712000	16398000	26105000
100	1	30000	1360000	8201000	1315000	16142000	27048000

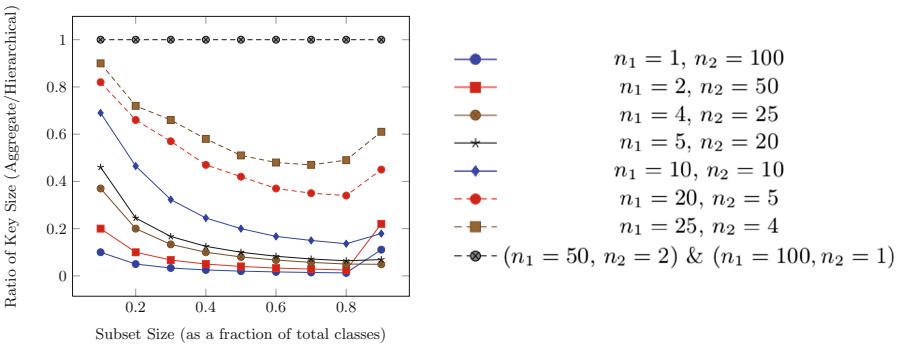

Fig. 3. Key size ratio - proposed aggregate scheme vs hierarchical scheme

7.3 Utilization Coefficient Comparison

Finally we compare the utilization-coefficient of the extended scheme for various values of n_1 and n_2 (corresponding to $n = 100$) with increase in the number of registered key pairs l, where each key pair increases the number of classes by n_2. We leave out the configuration $n_1 = n, n_2 = 1$ because that always leads to an utilization coefficient of 1 but is impractical due to huge space requirements. Figure 4 demonstrates that beyond a certain value of l, the combination $(1, n)$

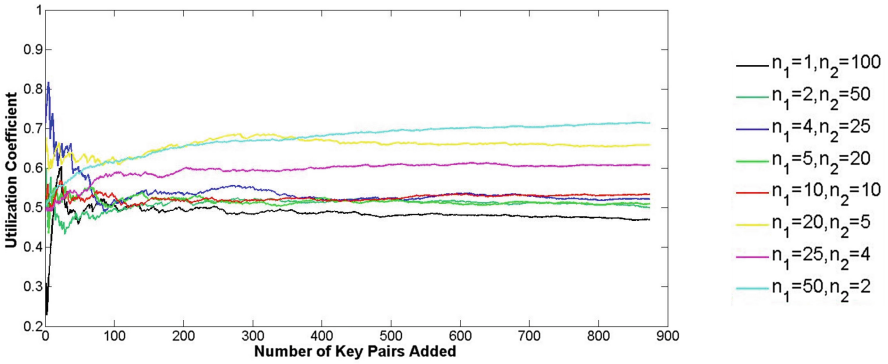

Fig. 4. Utilization coefficient vs newly registered keys

proposed in [3] has a lower utilization coefficient that all other combinations of (n_1, n_2) for a given n. This emphasizes the advantage of making the choice of (n_1, n_2) flexible.

8 Conclusions and Future Work

In this paper, we have proposed a secure and dynamic key aggregate encryption scheme for online data sharing. Our scheme allows data owners to delegate users with access rights to multiple ciphertext classes using a single decryption key that combines the decrypting power of individual keys corresponding to each ciphertext class. Unlike existing key aggregate schemes that are static in their access right delegation policies, our scheme allows data owners to dynamically revoke one or more users' access rights without having to change either the public or the private parameters/keys. The use of bilinear pairings over additive elliptic curve subgroups in our scheme helps achieve massive reductions in key and ciphertext sizes over existing schemes that use multiplicative groups. We pointed out that a possible criticism of this scheme is that the number of classes is predefined to some fixed n. To deal with this issue, we next proposed a generalized two-level construction of the basic scheme that runs n_1 instances of the basic scheme in parallel, with each instance handling key aggregation for n_2 ciphertext classes. This scheme provides two major advantages. First of all, it allows dynamic extension of ciphertext classes by registering of new public key-private key pairs without affecting other system parameters. Secondly, it provides a wide range of choices for n_1 and n_2 that allows efficient utilization of ciphertext classes while also achieving optimum space and time complexities. Finally, we extend the generalized scheme to allow the use of popular and efficiently implementable bilinear pairings in literature such as Tate Pairings that operate on multiple elliptic curve subgroups instead of one. Each of the three proposed schemes have been proven to be semantically secure. Experimental studies have demonstrated the superiority of our proposed scheme over existing ones in terms of key size

as well as efficient utilization of ciphertext classes. A possible future work is to make the proposed schemes secure against chosen ciphertext attacks.

References

1. Wang, C., Chow, S.S.M., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. Cryptology ePrint Archive, Report 2009/579 (2009). http://eprint.iacr.org/
2. Chow, S.S.M., Chu, C.-K., Huang, X., Zhou, J., Deng, R.H.: Dynamic secure cloud storage with provenance. In: Naccache, D. (ed.) Cryphtography and Security: From Theory to Applications. LNCS, vol. 6805, pp. 442–464. Springer, Heidelberg (2012)
3. Chu, C.-K., Chow, S.S.M., Tzeng, W.-G., Zhou, J., Deng, R.H.: Key-aggregate cryptosystem for scalable data sharing in cloud storage. IEEE Trans. Parallel Distrib. Syst. **25**(2), 468–477 (2014)
4. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
5. Ateniese, G., De Santis, A., Ferrara, A.L., Masucci, B.: Provably-secure time-bound hierarchical key assignment schemes. J. Cryptology **25**(2), 243–270 (2012)
6. Benaloh, J., Chase, M., Horvitz, E., Lauter, K.: Patient controlled encryption: ensuring privacy of electronic medical records. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, pp. 103–114. ACM (2009)
7. Ateniese, G., Kevin, F., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC) **9**(1), 1–30 (2006)
8. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
9. Silverman, J.H.: Advanced Topics in the Arithmetic of Elliptic Curves, vol. 151. Springer, New York (1994)
10. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Trans. Comput. Syst. (TOCS) **1**(3), 239–248 (1983)
11. Frey, G., Rück, H.-G.: A remark concerning-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comput. **62**(206), 865–874 (1994)
12. Hess, F., Smart, N.P., Vercauteren, F.: The eta pairing revisited. IEEE Trans. Inf. Theor. **52**(10), 4595–4602 (2006)
13. Zhao, C.-A., Zhang, F., Huang, J.: A note on the ate pairing. Int. J. Inf. Secur. **7**(6), 379–382 (2008)
14. Oliveira, L.B., Aranha, D.F., Morais, E., Daguano, F., López, J., Dahab, R.: Tinytate: computing the tate pairing in resource-constrained sensor nodes. In: 2007 Sixth IEEE International Symposium on Network Computing and Applications, NCA 2007, pp. 318–323. IEEE (2007)
15. Ghosh, S., Mukhopadhyay, D., Roychowdhury, D.: Secure dual-core cryptoprocessor for pairings over barreto-naehrig curves on FPGA platform. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **21**(3), 434–442 (2013)
16. Sandhu, R.S.: Cryptographic implementation of a tree hierarchy for access control. Inf. Process. Lett. **27**(2), 95–98 (1988)