Identity-Based Key Aggregate Cryptosystem from Multilinear Maps

Sikhar Patranabis and Debdeep Mukhopadhyay

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

Abstract. Secure online data sharing is one of the most important applications in today's world with far reaching impact in various fields. The advent of cloud computing has led to a massive requirement for secure, efficient, flexible and scalable data sharing schemes. A recent proposition in this regard is the *key-aggregate cryptosystem* (KAC). The main novelty of KAC is that it produces low overhead ciphertexts, and a low overhead decryption key that can aggregate the decryption rights to any arbitrarily large number of data classes. In this paper, we propose the first identity-based KAC where each data class can be associated with an unique identity. We present three different low-overhead KAC constructions based on $O(\log N)$-way multilinear maps that support N different data classes using polylogarithmic size public and private parameters. The constructions are proven to be secure under different security guarantees and are fully collusion-resistant against any number of colluding parties. We present some real-world applications where identity-based KAC proves to be a viable and efficient solution.
Keywords: Identity-based Key-Aggregate Cryptoystem, Online Data Sharing, Multilinear Maps, Collusion-resistant

1 Introduction

The recent advent of cloud computing has led to unforeseen amounts of data being shared online with wide-ranging applications. This has led to massive demands for online data sharing schemes that are flexible and scalable in terms of implementation, while also providing formal guarantees of security and resistance against multi-party collusion attacks. Storing encrypted data in a shared environment comes with two major overheads - efficient isolation of multiple classes of data, and secure delegation of decryption rights to different parts of this data to multiple users [DMMM$^+$12]. A trivial solution is to encrypt each class of data by a secret key, and then provide each user with decryption keys to specifically those set of data classes that she is authorized to access. This system is extremely inefficient since the number of private keys grows with the number of data classes. making it secure storage and distribution a massive challenge. Solutions based on fully homomorphic encryption (FHE), although theoretically sound, are currently too inefficient to be deployable in a practical environment.

A recently proposed public key based solution in this regard is the key-aggregate cryptosystem (KAC) [CCT$^+$14,PSM15]. Suppose a data owner wishes to share N different classes of data online, and wishes to grant decryption rights to an arbitrary subset $\mathcal{S}$ of these data classes to a user in an efficient manner. KAC allows her to achieve this by first encrypting each of her N different classes of data with separate public keys, and then combining the power of decryption for each class in $\mathcal{S}$ into a single low overhead aggregate key. The aggregate key can be distributed to the authorized users via a secure channel or using a public-key based broadcast encryption system [BGW05,BWZ14a]. The efficiency of any KAC construction is measured in terms of the *ciphertext size* (storage overhead) and the *aggregate key size* (distribution overhead). We say that a KAC construction has a low overhead if both the ciphertext overhead and the key aggregate overhead is upper bounded by a logarithmic function in the number of data classes N that the system can handle.

Low Overhead KAC Constructions in the Literature. Since KAC has only recently been introduced, there exist only a handful of constructions that achieve full collusion resistance while maintaining low ciphertext and aggregate key overhead. The first KAC proposed by Chu et. al. [CCT+14] achieves a *constant size* overhead for both the ciphertext and aggregate key, but provides neither formal guarantees of security nor proofs for collusion resistance. A more rigorous construction for KAC is provided by Patranabis et. al. in [PSM15] based on efficiently computable bilinear maps. Their scheme also achieves constant ciphertext and aggregate key overhead, and is proved to be CPA secure in the standard model. Additionally, their scheme is fully collusion resistant and is efficiently scalable to an arbitrary number of data owners. However, both the aforementioned constructions use a public parameter that has *linear* size in the number of data classes N. However, there are a number of open problems regarding KAC that have not been addressed prior to this work to the best of our knowledge.

- Constructing a KAC that is secure against an adaptive adversary in the standard model
- Constructing a KAC that is non-adaptively CCA secure in the standard model
- Constructing an identity-based KAC where each data class can be identified by a unique ID string

1.1 Our Contributions

In this paper, we propose three identity-based key-aggregate cryptosystems for N uniquely identifiable data classes using $O(\log N)$-way multilinear maps. We first describe each system using a basic single data owner case, and also demonstrate how they may be efficiently scaled to multi-data owner scenarios. The first two constructions presented in this paper have both ciphertext and aggregate decryption key overhead of $O(1)$ group elements, while the third has a ciphertext overhead of $O(\log N)$ group elements but the same aggregate key overhead of $O(1)$ group elements. Most importantly, the public parameter contains only $O(\log N)$ group elements in each construction, as opposed to $O(N)$ in the earlier KAC constructions [CCT+14,PSM15].

- The first KAC construction uses an asymmetric $O(\log N)$ multilinear map and is very similar to the basic KAC construction proposed in [PSM15]. The main contribution of this new scheme is the reduction in the size of the public parameters from $O(N)$ to $O(\log N)$ group elements, while maintaining low overhead for both the ciphertext and the aggregate key. The scheme is proved to be non-adaptively CPA and CCA secure under a standard complexity assumption introduced in [BWZ14a].
- The second KAC construction is based on a more general symmetric $O(\log N)$ multilinear map with similar overheads and space requirements as the first construction. The symmetric map setting allows us to obtain security proofs for non-adaptive CPA and CCA security based on a simpler complexity assumption. However, as a flip side, to maintain security we must ensure all data class indices $i \in \{1, \cdots, N\}$ can be efficiently mapped to integers $\hat{i} \in \{1, \cdots, O(N \log N)\}$, where all $\hat{i}$ have the same Hamming weight l.
- The third KAC construction mainly because it can be proved to be *adaptively* secure in generic multilinear groups using a tighter proof than the first two constructions. The tradeoff in this scheme, however, lies in the ciphertext size overhead, which is $O(\log N)$ group elements, unlike

$O(1)$ group elements in the previous constructions.

- Unlike existing KAC constructions, each of our proposed constructions are inherently *identity-based*, where each data class can be associated with a unique identity $id \in \{0,1\}^*$ with each data class. This is because all the parameters in our constructions have size at most polylogarithmic in the number of data classes N, N can now be as large as the range of a collision resistant hash function H. The index number for a particular class can be automatically set by hashing the corresponding class identity id to $H(id) \in \{1, \cdots, N\}$. Quite evidently, an identity-based KAC offers much greater flexibility and convenience of use.

1.2 Related Work

We present a brief overview of alternative 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 low overhead decryption keys for 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 [ADSFM12] 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 [BCHL09] 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. Efficient public key based encryption methods such as identity based encryption (IBE) [BF03] and attribute based encryption (ABE) [GPSW06] do not focus on key aggregation for multiple classes of data. Proxy re-encryption is another technique to achieve fine-grained access control and scalable user revocation in unreliable clouds [AFGH06]. 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 online data sharing applications.

2 Preliminaries

2.1 Key-Aggregate Cryptosystem (KAC)

We begin by formally defining the key-aggregate cryptosystem (KAC) framework. KAC is an ensemble of five randomized algorithms that are described next:

SetUp($\mathcal{ID}$): A data owner can classifying her data into one or more classes belonging an identity space $\mathcal{ID}$. The function sets up the key-aggregate cryptosystem for the identity space $\mathcal{ID}$. Outputs the public parameter $param$.

KeyGen(): Outputs the public key PK, the master-secret key msk and the authentication key U. A data owner willing to share her data using this system registers to receive her own public, private and authentication keys.

Encrypt($param, PK, i, \mathcal{M}$): Takes as input the public key parameter PK, the data class $i \in \mathcal{ID}$ and the plaintext message $\mathcal{M}$. Outputs the corresponding ciphertext $\mathcal{C}$, which is stored online in the shared environment.

Extract$(msk, \mathcal{S})$: Takes as input the master secret key and a subset of data classes $\mathcal{S} \subseteq \mathcal{ID}$. Computes the aggregate key $K_{\mathcal{S}}$ for all encrypted data/messages classified into any class in $\mathcal{S}$.

Decrypt$(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U)$: Takes as input the ciphertext $\mathcal{C}$, the data class i, the aggregate key $K_{\mathcal{S}}$ corresponding to a subset $\mathcal{S}$, and the authentication key U. If $i \notin \mathcal{S}$, output $\perp$. Otherwise, outputs the decrypted message $\mathcal{M}$.

Correctness. For correctness, we require that the decryption algorithm always succeeds in decrypting a correctly encrypted plaintext message m. Formally, correctness of KAC may be described as follows. For any valid identity space $\mathcal{ID}$, any set $\mathcal{S} \subseteq \mathcal{ID}$, any index $i \in \mathcal{S}$, and any plaintext message m, we must have

$$Pr[\mathbf{Decrypt}(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U) = \mathcal{M}|\mathcal{E}] = 1$$

where $\mathcal{E}$ is the event described as the conjunction of the following atomic events:

$$param \leftarrow \mathbf{SetUp}(\mathcal{ID}), (PK, msk, U) \leftarrow \mathbf{KeyGen}(),$$
$$\mathcal{C} \leftarrow \mathbf{Encrypt}(param, PK, i, \mathcal{M}), K_{\mathcal{S}} \leftarrow \mathbf{Extract}(msk, \mathcal{S})$$

2.2 Security Definitions

We define a formal framework for proving active chosen ciphertext security of KAC. We begin by introducing a game between a non-adaptive attack algorithm $\mathcal{A}$ and a challenger $\mathcal{B}$, both of whom are given $\mathcal{ID}$, the data class identity space, as input. The game proceeds through the following stages.

SetUp: Challenger $\mathcal{B}$ sets up the KAC system. In particular, $\mathcal{B}$ generates the public parameter $param$, the public key PK, the master secret key msk and the authentication key U.

Query Phase 1: Algorithm $\mathcal{A}$ adaptively issues decryption queries $q_1, \cdots, q_w$ where a decryption query comprises of the tuple $(\mathcal{C}, v)$, where $v \in \mathcal{ID}$ is the data class of the message encrypted as $\mathcal{C}$. The challenger has to respond with $\mathbf{Decrypt}(\mathcal{C}, v, \mathcal{S}, K_{\mathcal{S}}, U)$, for *any* $\mathcal{S} \subset \mathcal{ID}$ containing v.

Commit: Algorithm $\mathcal{A}$ adaptively commits to a set $\mathcal{S}^* \subset \mathcal{ID}$ of data classes that it wishes to attack. 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 data class $v \notin \mathcal{S}^*$. Next, $\mathcal{B}$ randomly chooses a data class $i \in \mathcal{S}^*$ and provides it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages 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 $\mathcal{C}^* = \mathbf{Encrypt}(PK, i, \mathcal{M}_b)$.

Query Phase 2: Algorithm $\mathcal{A}$ continues to adaptively issue decryption queries $q_{w+1}, \cdots, q_{Q_D}$ where a decryption query comprises of the tuple $(\mathcal{C}, v)$, but now subject to the restriction $\mathcal{C} \neq \mathcal{C}^*$. The challenger responds as in query phase 1.

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 to try and expose a message in this subset. We now formally define the security notions for KAC. Let $Adv_{\mathcal{A},|\mathcal{ID}|}$ denote the probability that $\mathcal{A}$ wins the game when the challenger is given $\mathcal{ID}$ as input.

Definition 2.1. A KAC construction is $(\epsilon, \mathcal{ID}, Q_D)$ adaptively secure under a chosen ciphertext attack (that is, adaptively CCA-secure) if, for all adaptive poly-time algorithms $\mathcal{A}$ that can make a total of Q_D decryption queries, we have that $|Adv_{\mathcal{A},|\mathcal{ID}|} - \frac{1}{2}| < \epsilon$.

Definition 2.2. A KAC construction is $(\epsilon, \mathcal{ID})$ adaptively secure under a chosen plaintext attack (that is, adaptively CPA-secure) if it is $(\epsilon, \mathcal{ID}, 0)$ adaptively CCA secure.

We also define two weaker notions of security in the non-adaptive setting. In particular, non-adaptive security is achieved in the scenario when $\mathcal{A}$ is required to commit to the set $\mathcal{S}^*$ before seeing the public parameters. We refer to such an adversary as a non-adaptive adversary. This leads to the following definitions.

Definition 2.3. A KAC construction is $(\epsilon, \mathcal{ID}, Q_D)$ non-adaptively secure under a chosen ciphertext attack (that is, non-adaptively CCA-secure) if, for all non-adaptive poly-time algorithms $\mathcal{A}$ that can make a total of Q_D decryption queries, we have that $|Adv_{\mathcal{A},|\mathcal{ID}|} - \frac{1}{2}| < \epsilon$.

Definition 2.4. A KAC construction is $(\epsilon, \mathcal{ID})$ non-adaptively secure under a chosen plaintext attack (that is, non-adaptively CPA-secure) if it is $(\epsilon, \mathcal{ID}, 0)$ non-adaptively CCA secure.

2.3 Multilinear Maps

In this section, we provide a brief overview of multilinear maps. Our description of multilinear maps is based on the *graded encoding scheme* used in several candidate multilinear map constructions [GGH13].

Symmetric Multilinear Maps. A standard symmetric multilinear map consists of the following pair of algorithms.

SetUp′$(1^\lambda, m)$: Sets up an m-linear map by outputting an m-tuple of groups $< \mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_m >$ of prime order q (where q is a λ bit prime), along with the respective generator $g_i \in \mathbb{G}_i$ for $1 \leq i \leq m$. In standard notation, $\mathbb{G}_1$ is the source group, $\mathbb{G}_m$ is the target group, and $\mathbb{G}_2, \cdots, \mathbb{G}_{m-1}$ are the intermediate groups.

$e_{i,j}(h_1, h_2)$: Takes as input $h_1 \in \mathbb{G}_i$ and $h_2 \in \mathbb{G}_j$, and outputs $h_3 \in \mathbb{G}_{i+j}$ such that

$$(h_1 = g_i^a, h_2 = g_j^b) \Rightarrow h_3 = g_{i+j}^{ab}$$

In this paper, we follow the standard notation used in the literature to omit the subscripts and simply refer to this multilinear map as e. Further, e may be generalized to multiple inputs as $e(h_1, \cdots, h_k) = e(h_1, e(h_2, \cdots, h_k))$. Note that g_i^a is sometimes referred to as the level-i *encoding* of a. The scalar a itself may therefore be referred to as the level 0 encoding of itself.

Asymmetric Multilinear Maps. We adopt the same definition of asymmetric multilinear maps presented in [GGH13]. According to this definition, in asymmetric multilinear maps, the groups are indexed by integer vectors. Formally, a standard asymmetric multilinear map consists of the following algorithms.

$\mathbf{SetUp}''(1^\lambda, \mathbf{m})$: Takes as input $\mathbf{m} \in \mathbb{Z}^l$. Sets up an $\mathbf{m}$-linear map by outputting an m-tuple of groups $< \mathbb{G}_{\mathbf{1}}, \mathbb{G}_{\mathbf{2}}, \cdots, \mathbb{G}_{\mathbf{m}} >$ of prime order q (where q is a λ bit prime), along with the respective generator $g_{\mathbf{v}} \in \mathbb{G}_{\mathbf{v}}$ for $\mathbf{1} \leq \mathbf{v} \leq \mathbf{m}$(comparison is defined component-wise). Further, let $\mathbf{x}_i$ be the ith *standard* basis vector (with 1 at position i and 0 at each other position). In standard notation, $\mathbb{G}_{\mathbf{x}_i}$ is the ith source group, $\mathbb{G}_{\mathbf{v}}$ is the target group, and the rest are the intermediate groups.

$e_{\mathbf{i},\mathbf{j}}(h_1, h_2)$: Takes as input $h_1 \in \mathbb{G}_{\mathbf{i}}$ and $h_2 \in \mathbb{G}_{\mathbf{j}}$, and outputs $h_3 \in \mathbb{G}_{\mathbf{i}+\mathbf{j}}$ such that

$$(h_1 = g_{\mathbf{i}}^a, h_2 = g_{\mathbf{j}}^b) \Rightarrow h_3 = g_{\mathbf{i}+\mathbf{j}}^{ab}$$

Again, we omit the subscripts and simply refer to this multilinear map as e, which may be generalized to multiple inputs as $e(h_1, \cdots, h_k) = e(h_1, e(h_2, \cdots, h_k))$.

In the forthcoming discussions, we present our KAC constructions assuming that the ideal multilinear maps based on the graded encoding scheme described above exist and are efficiently computable. We do this to make the analysis simple and easy to follow. We point out, however, that current candidates for multilinear maps in the cryptographic literature deviate from these ideal notions. In these candidates, group elements lack unique representations due to the presence of a noise term that tends to grow with repeated group/multilinear operations. We summarize the major properties of multilinear maps that our proposed KAC schemes require:

- The representation of an element should be statistically independent of the group and multilinear operations that led to that element.

- It should be possible to extract a *canonical* representation of an element in the target group given any representation of that element.

- The party setting up the multilinear map should have a *trapdoor* information that allows her to compute g^{α^x} for a non-random α and exponentially large x.

- It should be possible to generate asymmetric multilinear maps for any positive integer vector $\mathbf{m} \in \mathbb{Z}^l$.

- It should be possible to design the parameters of our system such that the noise growth during the execution of our scheme does not lead to erroneous computations.

We point out that the two foremost candidates for multilinear maps based on graded encoding schemes - the GGH candidate over ideal lattices [GGH13] and the CLT candidate over integers [CLT13] would allow us to meet all these requirements. However, we also note that both these candidate constructions have been subjected to *zeroizing attacks*[CHL+15], also known as the weak discrete logarithmic attack. These attacks break the Subgroup Membership (SubM) and the decision linear (DLIN) problems on the GGH candidate map, and also completely break the CLT candidate. Initially it was conjectured that this attack could be thwarted by keeping the low-level encodings of 0 private in the candidate constructions, and several fixes to these candidate constructions were provided based on this idea [GHMS14,BWZ14b]. However, these extensions were also proven to be insecure in [CLT14].

In this paper, we propose using use the graph-induced multilinear map based on lattices proposed by Gentry et al. [GGH15] to instantiate our constructions. The graph-induced multilinear map, like the GGH and CLT candidate constructions, is also based on the graded encoding scheme and meets almost all the requirements listed above. The only significant drawback of this construction is the absence of the *re-randomization* procedure (that helps to hide the group and multilinear operations leading to a particular element) to thwart cryptanalytic threats. However, a work around suggested in [GGH15] is to use Kilian-style randomization [Kil88] on the encoding side. This enhances the security of any scheme based on the graph induced candidate map, at the cost of some extra encoding bits.

3 KAC Using Asymmetric Multilinear Maps

In this section, we present the first construction of identity-based KAC based on asymmetric multilinear maps. Our construction is based on the basic KAC using bilinear pairings described in [PSM15]. Their construction involves outputting a public parameter set consisting of $O(N)$ group elements, where N is the number of data classes. Our goal in this scheme is to shrink the size of the public parameter to $O(\log N)$ group elements. To achieve this, we embed the original KAC scheme within a multilinear map, such that the original parameters can be derived from a small number of elements in the source group of the map. Hence it suffices to store these new elements as the public key of our proposed construction.

The Basic Idea. Let $N = 2^m - 1$ for some integer m, and let $\mathbf{m}$ be the $m+1$ length vector consisting of all ones. We use an asymmetric multilinear map with the target group $\mathbb{G}_{\mathbf{2m}}$. Note that if we pair two elements in the group $\mathbb{G}_{\mathbf{m}}$, we get an element in $\mathbb{G}_{\mathbf{2m}}$ by the definition of asymmetric multilinear maps. Let $Y_i = g_{\mathbf{m}}^{\alpha^i}$, where $\alpha \in \mathbb{Z}_q$. Recall that $\mathbf{x}_j$ is the jth *standard* basis vector (with 1 at position j and 0 at each other position) and $\mathbb{G}_{\mathbf{x}_j}$ is the jth source group with generator $g_{\mathbf{x}_i}$. Also, let $X_j = g_{\mathbf{x}_j}^{\alpha^{(2^j)}}$ for $0 \leq j \leq m-1$ and $X_m = g_{\mathbf{x}_m}^{\alpha^{(2^m+1)}}$. We make the following claims.

Claim 3.1. Given an i such that $0 \leq i \leq N$, Y_i can be computed from the set of parameters $(X_0, \cdots, X_m)$.

Proof. Let $i = \sum_{j=0}^{m-1} i_j 2^j$. We have

$$Y_i = e(X_0^{i_0} g_{\mathbf{x}_0}^{1-i_0}, \cdots, X_{m-1}^{i_{m-1}} g_{\mathbf{x}_{m-1}}^{1-i_{m-1}}, g_{\mathbf{x}_m})$$

Claim 3.2. Given i such that $N+2 \leq i \leq 2N$, Y_i can be computed from the set of parameters $(X_0, \cdots, X_m)$.

Proof. Let $i' = i - (2^m + 1) = \sum_{j=0}^{m-1} i'_j 2^j$. Then, we have

$$Y_i = e(X_0^{i'_0} g_{\mathbf{x}_0}^{1-i'_0}, \cdots, X_{m-1}^{i'_{m-1}} g_{\mathbf{x}_{m-1}}^{1-i'_{m-1}}, X_m)$$

We now make the following important observation.

Observation 3.3. *Unless $g_{\mathbf{x}_m}^{\alpha^{(2^m)}}$ is published, it is difficult to compute the value of Y_{N+1}.*

This is the basic trick we use to embed a parameter set comprising of $O(N)$ group elements into another parameter set comprising of $O(\log N)$ group elements. We next present the construction of the basic single data-owner KAC using this framework.

Assumption 3.4. For simplicity, we assume in the forthcoming discussion that our plaintext messages are embedded as elements in the group $\mathbb{G}_{\mathbf{2m}}$. We discuss in Section A how we may modify our scheme to relax this assumption.

3.1 Construction

Assume that $\mathbf{SetUp}''(1^\lambda, \mathbf{m})$ is the setup algorithm for an asymmetric multilinear map, where groups have prime order q (where q is a λ bit prime) and $\mathbb{G}_{\mathbf{m}}$ is the target group. Our first basic identity-based KAC, for a single data owner with $N = 2^m - 1$ data classes, consists of the following algorithms.

$\mathbf{SetUp}(1^\lambda, m)$: Take as input the length m of identities and the group order parameter λ. Set $\mathcal{ID} = \{0,1\}^m \backslash \{0\}^m$ as the identity space. Let $\mathbf{m}$ be the $m+1$ length vector consisting of all ones. Also, let $param'' \leftarrow SetUp''(1^\lambda, \mathbf{2m})$ be the public parameters for a multilinear map, with $\mathbb{G}_{\mathbf{2m}}$ being the target group. Choose a random $\alpha \in \mathbb{Z}_q$. Set $X_j = g_{\mathbf{x}_j}^{\alpha^{(2^j)}}$ for $0 \le j \le m-1$ and $X_m = g_{\mathbf{x}_m}^{\alpha^{(2^m+1)}}$. Output the public parameter tuple $param$ as

$$param = (param'', \{X_j\}_{j \in \{0, \cdots, m\}})$$

Discard α after $param$ has been output.

$\mathbf{KeyGen}()$: Randomly pick $\gamma, t \in \mathbb{Z}_q$. Set the master secret key msk to γ. Let $PK = g_{\mathbf{m}}^{\gamma}$. Finally set the user authentication key $U = g_{\mathbf{m}}^{t}$. Output the tuple (msk, PK, U).

$\mathbf{Encrypt}(PK, i, \mathcal{M})$: Take as input a message $\mathcal{M} \in \mathbb{G}_{\mathbf{2m}}$ belonging to class $i \in \mathcal{ID}$. Randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Recall that $Y_i = g_{\mathbf{m}}^{\alpha^i}$ and can be computed as per the formulation in Claim 3.1 for $1 \le i \le N$. Output the ciphertext $\mathcal{C}$ as

$$\mathcal{C} = (g_{\mathbf{m}}^{r}, (PK.Y_i)^{t'}, \mathcal{M}.g_{\mathbf{2m}}^{t'\alpha^{(2^m)}})$$

where $g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$ is computed as $(e(Y_{2^m-1}, Y_1))^{t'}$.

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

$$K_{\mathcal{S}} = \prod_{v \in \mathcal{S}} Y_{2^v - j}^{\gamma}$$

Note that this is indirectly equivalent to setting $K_{\mathcal{S}}$ to $\prod_{v \in \mathcal{S}} PK^{\alpha^{2^m - v}}$.

$\mathbf{Decrypt}(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U)$: If $i \notin \mathcal{S}$, output $\perp$. Otherwise, set

$$a_{\mathcal{S}} = \left(\prod_{v \in \mathcal{S}, v \neq i} Y_{2^m - v + i} \right) \text{ and } b_{\mathcal{S}} = \left(\prod_{v \in \mathcal{S}} Y_{2^m - v} \right)$$

Let $\mathcal{C} = (c_0, c_1, c_2)$. Output the decrypted message as

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

Correctness. To see that the scheme is correct, that is, $\hat{\mathcal{M}} = \mathcal{M}$, put $c_0 = g_{\mathbf{m}}^r$, $c_1 = (PK.Y_i)^{t'}$ and $c_2 = \mathcal{M}.g_{\mathbf{2m}}^{t\alpha^{(2^m)}}$. Then we have

$$\begin{aligned}
\hat{\mathcal{M}} &= c_2 \frac{e(K_{\mathcal{S}}.a_{\mathcal{S}}, U.c_0)}{e(b_{\mathcal{S}}, c_1)} \\
&= c_2 \frac{e(\prod_{v \in \mathcal{S}} Y_{2^m - v}^{\gamma} \cdot \prod_{v \in \mathcal{S}, v \neq i} Y_{2^m - v + i}, g_{\mathbf{m}}^{t'})}{e(\prod_{v \in \mathcal{S}} Y_{2^m - v}, (PK.Y_i)^{t'})} \\
&= c_2 \frac{e(\prod_{v \in \mathcal{S}, v \neq i} Y_{2^m - v + i}, g_{\mathbf{m}}^{t'})}{e(\prod_{v \in \mathcal{S}} Y_{2^m - v}, Y_i^{t'})} \\
&= \frac{\mathcal{M}.g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}}{e(Y_{2^m}, g_{\mathbf{m}}^{t'})} \\
&= \mathcal{M}
\end{aligned}$$

Implementation Nuances. As stated in Section 2, the only multilinear map candidate in the current literature that is not yet broken to the best of our knowledge is the graph-induced multilinear map construction proposed in [GGH15]. This graded-level encoding based construction contains noise terms that could lead to erroneous group operations, especially during repeated pairing computations. This could create complications, for example, in the computation of $g_{\mathbf{x}_j}^{\alpha^{2^j}}$ for sufficiently high values of j, especially if one attempts to compute it via level-0 encodings of random unknown α. However, a work-around for this is to pre-compute the level-0 encodings for the various α^{2^j} (where α is known) and then pair them with the corresponding $g_{\mathbf{x}_j}$. This means that the system administrator must herself set up the multilinear map framework with the knowledge of the secret parameters used to set up the multilinear maps. Note, however, that the knowledge of these paraameters is not required for either encryption and decryption, and hence may be discarded immediately after setup. Thus our KAC scheme may easily be instantiated by any noisy non-ideal candidate multilinear map without affecting the desired semantics in any way. Also note that the ciphertext and the aggregate key must not leak any important information, and hence need to be randomized appropriately. This implies that Kilian-style randomization parameters must be included for the group $\mathbb{G}_{\gg}$. No other randomization parameters are necessary. We now look into the security of our proposed identity-based KAC construction.

3.2 The Complexity Assumption

We now briefly state the complexity assumption that is to be used to prove the security of the proposed KAC scheme. The assumption is introduced in [BWZ14a].

The Hybrid Diffie-Hellman Exponent Assumption. Let $param''$ is generated by $\mathbf{SetUp}''(1^\lambda, \mathbf{2m})$, where $\mathbf{m}$ is the $m+1$ length vector consisting of all ones. Choose $\alpha \in \mathbb{Z}_q$ at random (where q is a λ-bit prime), and let $X_j = g_{\mathbf{x}_j}^{\alpha^{(2^j)}}$ for $0 \leq j \leq m-1$. Also, define $X_m = g_{\mathbf{x}_m}^{\alpha^{(2^m+1)}}$. Choose a random $t' \in \mathbb{Z}_q$, and let $V = g_{\mathbf{m}}^{t'}$. The decisional m-Hybrid Diffie Hellman Exponent (HDHE) problem as defined as follows. Given the tuple $(params'', \{X_j\}_{j \in \{0, \cdots, m\}}, V, Z)$, distinguish if Z is $g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$ or a random element of $\mathbb{G}_{\mathbf{2m}}$.

Definition 3.5. The decisional m-Hybrid Diffie-Hellman Exponent assumption holds for SetUp$''$ if, for any polynomial m and a probabilistic poly-time algorithm $\mathcal{A}$, $\mathcal{A}$ has negligible advantage in solving the m-Hybrid Diffie-Hellman Exponent problem.

3.3 Security of the Proposed KAC

We state and prove the non-adaptive CPA security of our proposed KAC scheme.

Theorem 3.6. *Let **Setup**$''$ be the setup algorithm for an asymmetric multilinear map, and let the decisional m-Hybrid Diffie-Hellman Exponent assumption holds for SetUp$''$. Then our proposed basic KAC for N data classes presented in Section 3.1 is non-adaptively CPA secure for $N = 2^m - 1$.*

Proof. Let $\mathcal{A}$ be a poly-time adversary such that $|Adv_{\mathcal{A},N} - \frac{1}{2}| > \epsilon$ for the proposed KAC system parameterized with an identity space $\mathcal{ID}$ of size $N = 2^m - 1$. Here ϵ is a non-negligible positive constant. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the decisional m-HDHE problem for **Setup**$''$. $\mathcal{B}$ takes as input a random m-HDHE challenge $(params'', \{X_j\}_{j\in\{0,\cdots,m\}}, V, Z)$ where:

- $param'' \leftarrow SetUp''(1^\lambda, 2\mathbf{m})$
- $X_j = g_{\mathbf{x}_j}^{\alpha^{(2^j)}}$ for $0 \leq j \leq m-1$
- $X_m = g_{\mathbf{x}_m}^{\alpha^{(2^m+1)}}$
- $V = g_{\mathbf{m}}^{t'}$ for a random $t' \in \mathbb{Z}_q$ (q being a λ bit prime)
- Z is either $g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$ or a random element of $\mathbb{G}_{\mathbf{2m}}$

$\mathcal{B}$ then proceeds as follows.

Commit: $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}$ of data classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a data class $i \in \mathcal{S}$ and provides it to $\mathcal{A}$.

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 $\mathcal{A}$. They are generated as follows.

- $param$ is set as $(param'', \{X_i\}_{i\in\{0,\cdots,m\}})$.
- PK is set as $g_{\mathbf{m}}^u / Y_i$ where u is chosen uniformly at random from $\mathbb{Z}_q$ and Y_i is computed as mentioned in Claim 3.1. Note that this is equivalent to setting $msk = u - \alpha^i$.
- U is set as $g_{\mathbf{m}}^t$ where t is again chosen uniformly at random from $\mathbb{Z}_q$.
- $\mathcal{B}$ then computes

$$K_{\overline{\mathcal{S}}} = \prod_{v \notin \mathcal{S}} \frac{Y_{2^m-v}^u}{Y_{2^m-v+i}}$$

  Observe that $K_{\overline{\mathcal{S}}} = \prod_{v\notin\mathcal{S}} PK^{\alpha^{2^m-v}}$, as desired. Moreover, $\mathcal{B}$ is aware that $i \notin \overline{\mathcal{S}}$ (implying $i \neq v$), and hence has all the resources to compute $K_{\overline{\mathcal{S}}}$.

Since the $g_{\mathbf{m}}$, α, u, t' and t values are chosen uniformly at random, *the public parameters and the public, private and authentication keys 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 $\mathbb{G}_{\mathbf{2m}}$, and provides them to $\mathcal{B}$. $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge as $(\mathcal{C}, \mathcal{M}_0, \mathcal{M}_1)$, where

$$\mathcal{C} = (U^{-1}V, V^u, \mathcal{M}_b.Z)$$

We claim that when $Z = g_{\mathbf{2m}}^{t\alpha^{(2^m)}}$ (i.e. the input to $\mathcal{B}$ is a valid m-HDHE 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, let $r = t' - t$. Then we have

$$U^{-1}V = g_{\mathbf{m}}^{r} \text{ and } V^u = (g_{\mathbf{m}}^{u})^{t'} = (PK.Y_i)^{t'}$$
$$\mathcal{M}_b.Z = \mathcal{M}_b.g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$$

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 $Z = g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$). Otherwise, it outputs 1 (indicating that Z is a random element in $\mathbb{G}_{\mathbf{2m}}$).

We conclude that $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$, which must therefore be negligible, as desired. This completes the proof of Theorem 3.6. Note that this proof is in the standard model and does not use random oracles. $\square$

3.4 CCA Secure Identity-Based KAC

We now demonstrate how to extend the basic identity-based KAC construction to obtain non-adaptive chosen ciphertext security while maintaining full collusion resistance. We have the following additional requirements for achieving CCA security:

- 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.

The CCA-Secure Construction. It is to be noted that unlike non-adaptive CPA security, non-adaptive CCA security for our proposed KAC under the m-HDHE assumption requires that the system handles at most $N-1$ data classes, where $N = 2^m - 1$. The reason for this will be apparent in the proof. hence for consistency of notation, we describe here the construction of the CCA-secure KAC for $N-1$ data classes. Recall that $\mathbf{SetUp}''(1^\lambda, \mathbf{m})$ is the setup algorithm for an asymmetric multilinear map, where groups have prime order q (where q is a λ bit prime).

$\mathbf{SetUp}(1^\lambda, m)$: Takes as input the length m of identities and the group order parameter λ. Let $\mathcal{ID} = \{0,1\}^m \backslash (\{0\}^m \cup \{1\}^m)$ be the data class identity space with $N-1$ classes. Also, let $\mathbf{m}$ be the $m+1$ length vector consisting of all ones. Let $param'' \leftarrow SetUp''(1^\lambda, \mathbf{2m})$ be the public parameters for a multilinear map, with $\mathbb{G}_{\mathbf{2m}}$ being the target group. Next, choose a random $\alpha \in \mathbb{Z}_q$. Set $X_j = g_{\mathbf{x}_j}^{\alpha^{(2^j)}}$ for $0 \leq j \leq m-1$ and $X_m = g_{\mathbf{x}_m}^{\alpha^{(2^m+1)}}$. Output the public parameter tuple $param$ as

$$param = (param'', \{X_j\}_{j \in \{0, \cdots, m\}})$$

Discard α after $param$ has been output.

KeyGen(): Same as in the construction 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 $V_{SIG} \in \mathbb{Z}_q$. Randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Recall that $Y_i = g_{\mathbf{m}}^{\alpha^i}$ and can be computed as per the formulation in Claim 3.1 for $1 \leq i \leq N$. Compute

$$\mathcal{C}' = (g_{\mathbf{m}}^r, (PK.Y_i.Y_{2^m-1}^{V_{SIG}})^{t'}, \mathcal{M}.g_{\mathbf{2m}}^{t'\alpha^{(2^m)}})$$

and output the ciphertext as

$$\mathcal{C} = (\mathcal{C}', Sign(\mathcal{C}', K_{SIG}), V_{SIG})$$

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

Decrypt$(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U)$: Let $\mathcal{C} = ((c_0, c_1, c_2), \sigma, V_{SIG})$. Verify that σ is a valid signature of (c_0, c_1, c_2) under the key V_{SIG}. If not, output $\perp$. Also, if $i \notin \mathcal{S}$, output $\perp$. Otherwise, set

$$SIG_{\mathcal{S}} = \prod_{v \in \mathcal{S}} Y_{2^{m+1}-1-v}^{V_{SIG}}$$
$$a_{\mathcal{S}} = \prod_{v \in \mathcal{S}, v \neq i} Y_{2^m-v+i}$$
$$b_{\mathcal{S}} = \prod_{v \in \mathcal{S}} Y_{2^m-v}$$

Note that these can be computed as $1 \leq i, v \leq N - 1 (= 2^m - 2)$. *This is precisely why we allow only $N - 1$ data classes.* Next, pick a random $w \in \mathbb{Z}_q$ and set

$$\hat{d_1} = (K_{\mathcal{S}}.SIG_{\mathcal{S}}.a_{\mathcal{S}}.(PK.Y_i.Y_{2^m-1}^{V_{SIG}})^w)$$
$$\hat{d_2} = (b_{\mathcal{S}}.g_{\mathbf{m}}^w)$$

Output the decrypted message

$$\hat{\mathcal{M}} = c_2 \frac{e(\hat{d_1}, U.c_0)}{e(\hat{d_2}, c_1)}$$

The proof of correctness of this scheme is presented in Appendix B.1. Note that the overhead for the ciphertext, aggregate key, public parameters, and the private and public keys, remains unchanged. The main change from the original scheme is in the fact that decryption requires a randomization value $w \in \mathbb{Z}_q$.

Claim 3.7. *For a given $i \in \mathcal{S}$, the pair $(\hat{d_1}, \hat{d_2})$ is chosen from the following distribution*

$$\left((Y_{2^m})^{-1} . \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^x, (g_{\mathbf{m}})^x\right)$$

where x is uniformly randomly chosen from $\mathbb{Z}_q$.
Proof. See Appendix B.2.

This randomization slows down the decryption by a factor of two, but is vital from the point of view of CCA-security. Note that the distribution $(\hat{d_1}, \hat{d_2})$ depends *only on the data class i for the message $\mathcal{M}$ to be decrypted* and is *completely independent of the subset $\mathcal{S}$ used to encrypt it.*

CCA Security. We next prove the non-adaptive CCA security of this scheme. Note that a signature scheme $(SigKeyGen, Sign, Verify)$ is said to be (ϵ, q_S) strongly existentially unforgeable if no poly-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 [CHK04].

Theorem 3.8. *Let **Setup**$''$ be the setup algorithm for an asymmetric multilinear map, and let the decisional m-Hybrid Diffie-Hellman Exponent assumption holds for SetUp$''$. Also, assume that the signature scheme is strongly existentially unforgeable. Then the modified KAC construction for $N-1$ data classes presented in Section 3.4 is non-adaptively CCA secure.*

Proof. Once again, let $\mathcal{A}$ be a poly-time adversary such that $|Adv_{\mathcal{A},N-1} - \frac{1}{2}| > \epsilon_1 + \epsilon_2$ for the proposed KAC system parameterized with an identity space $\mathcal{ID}$ of size $N-1 = 2^m - 2$. Let the signature scheme is (ϵ_2, q_S) strongly existentially unforgeable. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ_1 in solving the decisional m-HDHE problem for **Setup**$''$. $\mathcal{B}$ takes as input a random m-HDHE challenge $(params'', \{X_j\}_{j\in\{0,\cdots,m\}}, V, Z)$ where:

- $param'' \leftarrow SetUp''(1^\lambda, 2\mathbf{m})$
- $X_j = g_{\mathbf{x}_j}^{\alpha^{(2^j)}}$ for $0 \le j \le m-1$
- $X_m = g_{\mathbf{x}_m}^{\alpha^{(2^m+1)}}$
- $V = g_{\mathbf{m}}^{t'}$ for a random $t' \in \mathbb{Z}_q$, q being a λ bit prime
- Z is either $g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$ or a random element of $\mathbb{G}_{\mathbf{2m}}$

$\mathcal{B}$ then proceeds as follows.

Commit: $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}^*$ of data classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a data class $i \in \mathcal{S}^*$ and provides it to $\mathcal{A}$.

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 $V_{SIG}^* \in \mathbb{Z}_q$. The various items to be provided to $\mathcal{A}$ are generated as follows.

- $param$ is set as $(param'', \{X_i\}_{i\in\{0,\cdots,m\}})$.
- PK is set as $(g_{\mathbf{m}}^u) / \left(Y_i . Y_{2^m-1}^{V_{SIG}^*}\right)$ where u is chosen uniformly at random from $\mathbb{Z}_q$ and Y_i, Y_{2^m-1} are computed as mentioned in Claim 3.1. Note that this is equivalent to setting $msk = u - \alpha^i - V_{SIG}^* \alpha^{2^m-1}$, as required.
- U is set as $g_{\mathbf{m}}^t$ where t is again chosen uniformly at random from $\mathbb{Z}_q$.
- $\mathcal{B}$ then computes
$$K_{\overline{\mathcal{S}^*}} = \prod_{v \notin \mathcal{S}^*} \frac{Y_{2^m-v}^u}{(Y_{2^m-v+i}).(Y_{2^{m+1}-1-v}^{V_{SIG}^*})}$$
Observe that $K_{\overline{\mathcal{S}^*}} = \prod_{v\notin\mathcal{S}^*} PK^{\alpha^{2^m-v}}$, as desired. Moreover, $\mathcal{B}$ is aware that $i \notin \overline{\mathcal{S}^*}$ (implying $i \neq v$), and hence has all the resources to compute $K_{\overline{\mathcal{S}^*}}$.

Since the $g_{\mathbf{m}}$, α, u, t' and t values are chosen uniformly at random, *the public parameters and the public, private and authentication keys have an identical distribution to that in the actual construction.*

Query Phase 1: Algorithm $\mathcal{A}$ now issues decryption queries. Let $(\mathcal{C}, v)$ be a decryption query $\mathcal{C}$ is obtained by $\mathcal{A}$ using some subset $\mathcal{S}$ containing v. However, $\mathcal{B}$ is not given the knowledge of $\mathcal{S}$. 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

$$\begin{aligned}
\hat{d}_0 &= Y_{2^m-1}^{(V_{SIG}-V_{SIG}^*)}.Y_v.Y_i^{-1} \\
\hat{d}'_0 &= (Y_{v+1}/Y_{i+1})^{\frac{1}{(V_{SIG}-V_{SIG}^*)}} \\
\hat{d}_2 &= g_{\mathbf{m}}^x.Y_1^{\frac{1}{(V_{SIG}-V_{SIG}^*)}} \\
\hat{d}_1 &= \left(\hat{d}_2\right)^u.\left(\hat{d}_0\right)^x.\left(\hat{d}'_0\right)
\end{aligned}$$

Note that $\hat{d}'_0$ can be computed following Claim 3.1 as $1 \leq i, v \leq 2^m - 2$. $\mathcal{B}$ responds with $K = c_2 \frac{e(\hat{d}_1, c_0.U)}{e(\hat{d}_2, c_1)}$.

Claim 3.9. *$\mathcal{B}$'s response is exactly as in a real attack scenario, that is, for some x' chosen uniformly at random from $\mathbb{Z}_q$, we have*

$$\hat{d}_1 = (Y_{2^m})^{-1}.\left(PK.Y_v.Y_{2^m-1}^{V_{SIG}}\right)^{x'} \text{ and } \hat{d}_2 = g_{\mathbf{m}}^{x'}$$

Proof. See Appendix B.3.

Thus, by the result in Claim 3.7, $\mathcal{B}$'s response is identical to **Decrypt**$(\mathcal{C}, v, \mathcal{S}, K_{\mathcal{S}}, U)$, even though $\mathcal{B}$ does not possess the knowledge of the subset $\mathcal{S}$ used by $\mathcal{A}$ to obtain $\mathcal{C}$.

Challenge: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages in $\mathbb{G}_{\mathbf{2m}}$, and provides them to $\mathcal{B}$. $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets

$$\begin{aligned}
\mathcal{C} &= (U^{-1}V, V^u, \mathcal{M}_b.Z) \\
\mathcal{C}^* &= (\mathcal{C}, Sign(\mathcal{C}, K_{SIG}^*), V_{SIG}^*)
\end{aligned}$$

The challenge posed to $\mathcal{A}$ is $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$. It can be easily shown that when $Z = g_{\mathbf{2m}}^{t\alpha^{(2^m)}}$ (i.e. the input to $\mathcal{B}$ is a valid m-HDHE tuple), then this 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 (indicating that $Z = g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}$). Otherwise, it outputs 1 (indicating that Z is a random element in $\mathbb{G}_{\mathbf{2m}}$).

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 secret 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, $Pr[\text{abort}] < \epsilon_2$.

We conclude that $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$, which must therefore be negligible, as desired. This completes the proof of Theorem 3.8. □

3.5 Generalized Multi-Data Owner Identity-Based KAC

The basic version of KAC discussed so far accommodates a single data owner with N data classes. In a practical deployment scenario, we would ideally want a system where multiple users can store their own data, and delegate users with decryption rights to their own data classes. The challenge in designing such a system is to ensure that data owned by different data owners should be isolated and the aggregate key for one data owner should not reveal the data for another data owner. This concept is referred to as *local aggregation*.

We tackle this issue by proposing a generalized KAC construction that uses the cryptographic security provided by the basic KAC and extends it to accommodate multiple users. The basic idea is to run several parallel instances of the basic KAC, one for each data owner. Each basic KAC can handle $N = 2^m - 1$ data classes. If a data owner wishes to store more than N classes of data, she may request for more basic KAC instances. We refer to each of the individual owner units as *data superclasses*. Each data class now has a double index (i_1, i_2) where i_1 is the super-class index that identifies the data owner, while i_2 is the sub-class index specific to the data owner. Each superclass i_1 is characterized by its own master secret key msk^{i_1}, public key PK^{i_1}, and authentication key U^{i_1}. However, the main advantage of this approach is that all the parallel instances can share the same public $param$, which needs to be setup exactly once by the system administrator. The system achieves local aggregation within each superclass in the sense that decryption rights to any subset of data classes in the same superclass can be aggregated into a single aggregate key with the same overhead as before. The ciphertext overhead corresponding to each plaintext message also remains unchanged.

The proposed construction is highly scalable to an arbitrary number of data owners, who can 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 still the m-HDHE assumption, and is not altered by the increase in the number of data classes.

4 KAC Using Symmetric Multilinear Maps

In this section, we present the second identity-based KAC construction based on traditional symmetric multilinear maps. We use the same idea presented in the earlier construction, that is, we embed the original KAC scheme in [PSM15] within a symmetric multilinear map, such that the original public parameters can be derived from a small number of elements in the source group of the map. In this construction, the parameter $Y_i = g_m^{\alpha^i}$, while $X_j = g_1^{\alpha^{2^j}}$. However, unlike in the asymmetric setting where the same elements cannot be paired together, in the symmetric setting one could pair X_{m-1} with itself, and then pair it with g_1 $m-2$ times, to obtain Y_{N+1}. To overcome this we use a technique proposed in [BWZ14a] that restricts the bit representations of all identities in $\mathcal{ID}$ to a single Hamming weight class. This actually allows computing the necessary Y_i without leaking the value of Y_{N+1}.

The Basic Idea. Let $Y_i = g_{m-1}^{\alpha^i}$ and $\hat{Y}_i = g_l^{\alpha^i}$ where $l \leq m$. Set $X_j = g_1^{\alpha^{(2^j)}}$ for $i = 0, 1, \cdots, m$. Further, let $HW(i)$ denote the Hamming weight of the bit representation of i. We now make the following claims.

Claim 4.1.a. One can compute $g_{HW(i)}^{\alpha^i}$ for $1 \leq i \leq 2^m - 2$. In particular, one can compute $\hat{Y}_i$ for $1 \leq i \leq 2^m - 2$ such that $HW(i) = l$.

Proof. Compute $g_{HW(i)}^{\alpha^i}$ by pairing together all X_j such that the jth bit of i is 1. Since i has at most m bits, the necessary X_j are available.

Claim 4.1.b. One can compute Y_i for $1 \leq i \leq 2^m - 2$.

Proof. Compute $g_{HW(i)}^{\alpha^i}$ by Claim 4.1.a. (and pair it with $g_{m-HW(i)-1}$ if $HW(i) \leq m-2$). Note that for all i such that $1 \leq i \leq 2^m - 2$, $HW(i) \leq m-1$.

Claim 4.2. One can compute Y_{2^m-1-i} for $1 \leq i \leq 2^m - 2$ such that $HW(i) = l$.

Proof. Compute $g_{HW(2^m-1-i)}^{\alpha^{2^m-1-i}}$ as per Claim 4.1.a. Note that $HW(2^m-1-i) = m-l$ if $HW(i) = l$. Thus, we basically computed $g_{m-l}^{\alpha^{2^m-1-i}}$. Then, we pair it with g_{l-1} (obtained by pairing g_1 $(l-1)$ times) to obtain Y_{2^m-1-i}.

Claim 4.3. One can compute $Y_{2^m-1-v+i}$ for $1 \leq i, v \leq 2^m - 2$, $i \neq v$ where $HW(i) = HW(v) = l$.

Proof. Let T_1 denote the set of these bit positions that are 1 in the bit representation of i, and T_2 denote the set of bit positions that are 1 in the bit representation of $2^m - 1 - v$. Clearly, $|T_1| = l$ and $|T_2| = m - l$. Now, note that that $T_1 \bigcap T_2 = \phi$ iff $i = v$ which is not allowed. So $\exists j' \in T_1 \bigcap T_2$. Then, we can write

$$2^m - 1 - v + i = \left(\sum_{j \in T_1 \setminus \{j'\}} 2^j \right) + \left(\sum_{j \in T_2 \setminus \{j'\}} 2^j \right) + 2^{j'+1}$$

Note that this is a sum of $m-1$ powers of two. This in turn allows us to compute

$$Y_{2^m-1-v+i} = e\left(\{X_j\}_{j \in T_1 \setminus \{j'\}}, \{X_j\}_{j \in T_2 \setminus \{j'\}}, X_{j'+1} \right)$$

which is a pairing of $(m-1)$ X_j terms, as desired.

Assumption 4.4. For simplicity, we assume in the forthcoming discussion that our plaintext messages are embedded as elements in the group $\mathbb{G}_{m+l-1}$. For relaxations, refer Section A.

4.1 Construction

We now present the basic construction of KAC using traditional symmetric multilinear maps. Recall that $\mathbf{SetUp}'(1^\lambda, m)$ sets up an m-linear map with groups of prime order q (q being a λ bit prime) and the target group $\mathbb{G}_m$. Our second identity-based KAC consists of the following algorithms.

$\mathbf{SetUp}(1^\lambda, (m, l))$: Set up the KAC system for $\mathcal{ID}$ consisting of all m bit class identities with Hamming weight l, that is $N = |\mathcal{ID}| = \binom{m}{l}$. Since $1 \leq l \leq m-1$, we have $N \leq 2^{m-2}$. Let $param' \leftarrow SetUp'(1^\lambda, m+l-1)$ be the public parameters for a symmetric multilinear map, with $\mathbb{G}_{m+l-1}$ being the target group. Choose a random $\alpha \in \mathbb{Z}_q$. Set $X_j = g_1^{\alpha^{(2^j)}}$ for $0 \leq j \leq m$. Output the public parameter tuple $param$ as

$$param = (param', \{X_j\}_{j \in \{0, \cdots, m\}})$$

Discard α after $param$ has been output.

$\mathbf{KeyGen}()$: Randomly pick $\gamma, t \in \mathbb{Z}_q$. Set the master secret key msk to γ. Let $PK = g_l^\gamma$. Finally set the user authentication key $U = g_l^t$. Output the tuple (msk, PK, U).

Encrypt$(PK, i, \mathcal{M})$: Take as input a message $\mathcal{M} \in \mathbb{G}_{m+l-1}$ belonging to class $i \in \mathcal{ID}$. Randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Recall that $\hat{Y}_i = g_l^{\alpha^i}$ and can be computed for $i \in \mathcal{ID}$ as per Claim 4.2. Output the ciphertext $\mathcal{C}$ as

$$\mathcal{C} = (g_l^r, (PK.\hat{Y}_i)^{t'}, \mathcal{M}.g_{m+l-1}^{t'\alpha^{(2^m-1)}})$$

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

$$K_{\mathcal{S}} = \prod_{v \in \mathcal{S}} (Y_{2^m-1-v})^{\gamma}$$

Recall that Y_{2^m-1-v} can be computed as per Claim 4.3 for $j \in \mathcal{ID}$. Note that this is indirectly equivalent to setting $K_{\mathcal{S}}$ to $\prod_{v \in \mathcal{S}} \left(g_{m-1}^{msk}\right)^{\alpha^{2^m-1-v}}$.

Decrypt$(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U)$: If $i \notin \mathcal{S}$, output $\perp$. Otherwise, use the results from Claims 4.4 and 4.3 to set

$$a_{\mathcal{S}} = \left(\prod_{v \in \mathcal{S}, v \neq i} Y_{2^m-1-v+i}\right) \text{ and } b_{\mathcal{S}} = \left(\prod_{v \in \mathcal{S}} Y_{2^m-1-v}\right)$$

Let $\mathcal{C} = (c_0, c_1, c_2)$. Output the decrypted message as

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

Correctness. Refer Appendix C.1.

Finally, we comment on the choice of m and l. Let $N = |\mathcal{ID}|$ be the number of classes our proposed KAC wishes to handle. Then we must have $N = \binom{m}{l}$. If we wish to minimize the value of m, we may set $m = \log N + \lceil(\log \log N)/2\rceil + 1$ and $l = \lfloor m/2 \rfloor$. But if we wish to minimize the degree of multilinearity, then we must set $m \approx 1.042(\log_2 N + (\log_2 \log_2 N)/2)$ and $l \approx 0.398(\log_2 N + (\log_2 \log_2 N)/2)$, leading to a total multilinearity requirement of $1.44(\lambda + (\log_2 \lambda)/2) - 1$ [BWZ14a].

Implementation Nuances. As in the earlier construction, the system administrator must herself set up the multilinear map framework with the knowledge of the secret parameters for setting up the map. Also, the ciphertext and the aggregate key must not leak any important information and hence need to be appropriately randomized. This implies that Kilian-style randomization parameters must be included for the groups $\mathbb{G}_l$ and $\mathbb{G}_{m-1}$. We now look into the security of our second identity-based KAC construction.

4.2 The Complexity Assumption

We now briefly state the complexity assumption that is to be used to prove the security of second KAC scheme.

The Multilinear Diffie-Hellman Exponent Assumption. Let $param'$ is generated by **SetUp**$'(1^\lambda, m+l-1)$. Choose $\alpha \in \mathbb{Z}_q$ at random (where q is a λ-bit prime), and let $X_j = g_1^{\alpha^{(2^j)}}$ for $0 \leq j \leq m$. Choose a random $t' \in \mathbb{Z}_q$, and let $V = g_l^{t'}$. The decisional (m, l)-Multilinear Diffie Hellman Exponent (MDHE) problem as defined as follows. Given the tuple $(params', \{X_j\}_{j \in \{0, \cdots, m\}}, V, Z)$, distinguish if Z is $g_{m+l-1}^{t'\alpha^{(2^m-1)}}$ or a random element of $\mathbb{G}_{m+l-1}$.

Definition 4.5. The decisional (m, l)-Multilinear Diffie-Hellman Exponent assumption holds for SetUp$'$ if, for any polynomial m and a probabilistic poly-time algorithm $\mathcal{A}$, $\mathcal{A}$ has negligible advantage in solving the m-Multilinear Diffie-Hellman Exponent problem.

4.3 Security of the Proposed KAC

We state and prove the non-adaptive CPA security of our proposed KAC scheme.

Theorem 4.6. *Let **Setup**$'$ be the setup algorithm for a symmetric multilinear map, and let the decisional (m, l)-Multilinear Diffie-Hellman Exponent assumption holds for SetUp$'$. Then our proposed construction of KAC for N data classes presented in Section 4.1 is non-adaptively CPA secure for $N = \binom{m}{l}$.*

Proof. Let $\mathcal{A}$ be a poly-time adversary such that $|Adv_{\mathcal{A},N} - \frac{1}{2}| > \epsilon$ for the proposed KAC system parameterized with an identity space $\mathcal{ID}$ of size N, where $N = \binom{m}{l}$ and ϵ is a non-negligible positive constant. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the decisional (m, l)-MDHE problem for **Setup**$'$. $\mathcal{B}$ takes as input a random (m, l)-MDHE challenge consisting of the tuple $(params', \{X_j\}_{j\in\{0,\cdots,m\}}, V, Z)$, where:

- $param' \leftarrow SetUp'(1^\lambda, m + l - 1)$
- $X_j = g_1^{\alpha^{(2^j)}}$ for $0 \le j \le m$
- $V = g_l^{t'}$ for a random $t' \in \mathbb{Z}_q$ (where q is a λ bit prime)
- Z is either $g_{m+l-1}^{t'\alpha^{(2^m-1)}}$ or a random element of $\mathbb{G}_{m+l-1}$.

$\mathcal{B}$ then proceeds as follows.

Commit: $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}$ of data classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a data class $i \in \mathcal{S}$ and provides it to $\mathcal{A}$.

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 $\mathcal{A}$. They are generated as follows.

- $param$ is set as $(param', \{X_i\}_{i\in\{0,\cdots,m\}})$.
- PK is set as $(\frac{g_l^u}{\hat{Y}_i})$ where u is chosen uniformly at random from $\mathbb{Z}_q$ and $\hat{Y}_i$ is computed as mentioned in Claim 4.2. Note that this is equivalent to setting $msk = u - \alpha^i$.
- U is set as g_l^t where t is again chosen uniformly at random from $\mathbb{Z}_q$.
- $\mathcal{B}$ then computes
$$K_{\overline{\mathcal{S}}} = \prod_{v \notin \mathcal{S}} \frac{(Y_{2^m-1-v})^u}{Y_{2^m-1-v+i}}$$
Observe that $K_{\overline{\mathcal{S}}} = \prod_{v\in\mathcal{S}} \left(g_{m-1}^{msk}\right)^{\alpha^{2^m-1-v}}$, as desired. Moreover, $\mathcal{B}$ is aware that $i \notin \overline{\mathcal{S}}$ (implying $i \neq v$), and hence has all the resources to compute $K_{\overline{\mathcal{S}}}$.

Since g_{m-1}, g_l, α, u, t' and t values are chosen uniformly at random, *the public parameters and the public, private and authentication keys 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 $\mathbb{G}_{m+l-1}$, and provides them to $\mathcal{B}$. $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets the challenge as $(\mathcal{C}, \mathcal{M}_0, \mathcal{M}_1)$, where
$$\mathcal{C} = (U^{-1}V, V^u, \mathcal{M}_b.Z)$$

We claim that when $Z = g_{m+l-1}^{t'\alpha^{(2^m-1)}}$ (i.e. the input to $\mathcal{B}$ is a valid (m,l)-MDHE 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, let $r = t' - t$ and refer the following relations.

$$U^{-1}V = g_l^r \text{ and } V^u = (g_l^u)^{t'} = (PK.\hat{Y}_i)^{t'}$$
$$\mathcal{M}_b.Z = \mathcal{M}_b.g_{m+l-1}^{t'\alpha^{(2^m-1)}}$$

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 $Z = g_{m+l-1}^{t'\alpha^{(2^m-1)}}$). Otherwise, it outputs 1 (indicating that Z is a random element in $\mathbb{G}_{m+l-1}$).

We conclude that $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$, which must therefore be negligible, as desired. This completes the proof of Theorem 4.7. Note that once again this proof is also in the standard model and does not use random oracles. $\square$

CCA Security. We can easily extend this KAC construction for non-adaptive CCA security as we did in Section 3.4 for full collusion resistance. For the detailed construction and security proof, refer Appendix C.2.

Generalization to Multi-Data Owner System. Following the techniques discussed in Section 3.5, our second KAC construction can also be scaled to an arbitrary number of data owners, who can 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 still the (m,l)-MDHE assumption, and is not altered by the increase in the number of data classes.

5 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 [Pau] and Dropbox [Clo] 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 wons 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 [BCHL09]. 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.

6 Security Under The Generic Multilinear Map Setting

Finally, we investigate the security of our proposed KAC schemes in the generic multilinear map model. We first give a brief overview of the model. We then review the security of our proposed KAC schemes under this model. Finally, we propose a third KAC construction that is adaptively secure under this model for much smaller values of the prime group order q.

6.1 Generic Multilinear Maps

Just as multilinear maps are an extension of bilinear maps, the generic multilinear map model is an extension of the generic bilinear map model [BBG05]. We describe the model here for completeness. In this model, the group $\mathbb{G}_{\succapprox}$ for $v \in \mathbb{Z}_l$ is represented by a random injective function $\xi : \mathbb{Z}_q \times \mathbb{Z}^l \rightarrow \{0,1\}^n$ [BWZ14a]. Suppose that the target vector is $\mathbf{n} \in \mathbb{Z}^l$. Any algorithm in the generic multilinear map model is said to interact with the map using the tuple of algorithms (**Encode**,**Mult**,**Pair**) described below.

Encode$(x, \mathbf{v})$: Takes as input a non-negative integer vector $\mathbf{v} \leq \mathbf{m}$ and outputs $\xi(x, \mathbf{v})$.

Mult$(\xi_1, \xi_2, \diamond)$: Takes as input $\xi_1 = \xi(x_1, \mathbf{v})$, $\xi_2 = \xi(x_2, \mathbf{v})$ and $\diamond \in \{+, -\}$. Outputs $\xi(x_1 \diamond x_2, \mathbf{v})$.

Pair(ξ_1, ξ_2): Takes as input $\xi_1 = \xi(x_1, \mathbf{v}_1)$ and $\xi_2 = \xi(x_2, \mathbf{v}_2)$ where $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v} \leq \mathbf{m}$. Outputs $\xi(x_1.x_2, \mathbf{v})$.

Note that if the inputs are not valid, each off the above algorithms returns $\perp$. Also, **Mult** and **Pair** here are assumed to be oracles to compute the induced group multiplication and multilinear map operations.

6.2 Security of Our KAC Constructions

Boneh, Boyen and Goh [BBG05] introduced a general technique to prove computational lower bounds on the difficulty of breaking Diffie-Hellman-type complexity assumptions in a generic bilinear group model. An extension of these techniques can be used to prove that the m-HDHE and the (m, l)-MDHE assumptions are hard in the generic multilinear map model. However, the presence of high degree exponents such as α^{2^m} in these assumptions means that the adversary can construct polynomials with degree as high as $m2^m$ in the secret α. As pointed out in [BWZ14a], this means the upper bound on the advantage of a generic adversary making at most t queries is only $\approx \left(t^2 2^m/q\right)$. This in turn implies that both non-adaptive and adaptive λ-bit security for our earlier KAC constructions demands a prime group order $q \approx 2^{3\lambda}$ instead of 2^λ. This motivates us to present an identity-based KAC construction that is adaptively secure in the generic multilinear map model for $q = 2^\lambda$.

7 An Adaptively Secure KAC in the Generic Multilinear Map Model

In this section, we present a fully collusion resistant key-aggregate cryptosystem that is adaptively CPA secure in the generic multilinear map model for standard group order parameter q.

The Basic Idea. As in the previous constructions, the idea is to somehow embed the $O(N)$ public parameters in a set of $O(\log N)$ parameters such that the overall resource requirement of the system is reduced from linear to at most polylogarithmic. However, this scheme is slightly different from the earlier constructions in the sense that the public parameters in this scheme are not derived from a single scalar α. In fact, each parameter is to be derived from a separate random scalar. The other main challenge presented by this scheme is that the ciphertext does not consist of a constant number of group elements, unlike in the previous constructions. However, again we need to ensure that the overhead remains bounded by $O(\log N)$ and does not blow up to $O(N)$. Thus we need to have a way to reduce both the public parameter and ciphertext size overhead. As demonstrated in the construction, we handle both of these requirements by resorting to the use of Naor-Reingold-style pseudorandom functions (PRFs) [NR04].

7.1 Construction

Let $Setup'(1^\lambda, m)$ be the setup algorithm for an m-linear map with groups of prime order q (q being a λ bit prime) and the target group $\mathbb{G}_m$. Our third and final identity-based KAC consists of the following algorithms.

SetUp$(1^\lambda, m)$: Set up the KAC system for $\mathcal{ID}$ consisting of all m bit class identities. Let $param' \leftarrow SetUp'(1^\lambda, m+1)$ be the public parameters for a symmetric multilinear map, with $\mathbb{G}_{m+1}$ being the target group. For $j = 0, \cdots, m-1$ and $b = 0, 1$, generate random $\alpha_{j,b} \in \mathbb{Z}_q$ and let $X_{j,b} = g_1^{\alpha_{j,b}}$. Output the public parameter as

$$param = (param', \{X_{j,b}\}_{j \in \{0,\cdots,m-1\}, b \in \{0,1\}})$$

Discard all $\alpha_{j,b}$ after $param$ has been output.

Claim 7.1. For any class index number $i \in \mathcal{ID}$, one can compute $Y_i = g_m^{\prod_{j=0}^{m-1} \alpha_{j,i_j}}$ where i_j is the jth bit in the binary representation of i.

Proof. Compute Y_i as $e(X_{0,i_0}, X_{1,i_1}, \cdots, X_{m-1,i_{m-1}})$. Note that each Y_i value is essentially the output of a Naor-Reingold-style PRF.

KeyGen(): Randomly pick $\gamma, x, t \in \mathbb{Z}_q$. Set $msk = (\gamma, x)$, $PK = (g_m^\gamma, g_1^x)$ and $U = g_1^t$. Output the tuple (msk, PK, U).

Encrypt$(PK, i, \mathcal{M})$: Take as input message $\mathcal{M}$ in class $i \in \mathcal{ID}$ and $PK = (PK_1, PK_2)$. Randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Set

$$\begin{aligned} c_0 &= g_1^r \\ c_{j+1} &= X_{j,(1-u_j)}^{t'} \text{ for } j = 0, 1, \cdots, m-1 \\ c_{m+1} &= PK_1.Y_i^{t'} \\ c_{m+2} &= \mathcal{M}.\left(g_{m+1}^{\gamma x}\right)^{t'} = \mathcal{M}.\left(e(PK_1, PK_2)\right)^{t'} \end{aligned} \tag{1}$$

Finally, output the ciphertext as

$$\mathcal{C} = (\{c_j\}_{j \in \{0, \cdots, m+2\}})$$

Claim 7.2. For any class index $v \neq i$, one can compute $Y_v^{t'}$ given $\mathcal{C}$.

Proof. Let $\mathcal{C} = (c_0, \cdots, c_{m+2})$. Since $v \neq i$, there exists a bit position $j \in \{0, \cdots, m-1\}$ such that $v_j \neq i_j$. This allows one to compute

$$\begin{aligned} e(X_{0,v_0}, \cdots, X_{j-1,v_j-1}, c_{j+1}, X_{j+1,v_j+1}, \cdots, X_{m-1,v_{m-1}}) &= g_m^{t' \alpha_{j,v_j} \cdot \prod_{j' \neq j} \alpha_{j',v_{j'}}} \\ &= g_m^{t' \cdot \prod_{j'=0}^{m-1} \alpha_{j',v_{j'}}} \\ &= Y_v^{t'} \end{aligned}$$

Note that for a given i, each $Y_v^{t'}$ is the output of a puncturable Naor-Reingold style PRF. In particular, the PRF is punctured at i to generate the value of $Y_v^{t'}$ for $v \neq i$, without requiring the knowledge of $Y_i^{t'}$.

Extract$(msk, \mathcal{S})$: Let the input $msk = (msk_1, msk_2)$. For the input subset of data class indices $\mathcal{S}$, the aggregate key is computed as

$$K_{\mathcal{S}} = \left(\prod_{v \in \mathcal{S}} Y_v\right)^{msk_2}$$

Decrypt$(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U, PK)$: We slightly alter the semantics of the **Decrypt** operation in the sense that it also takes $PK = (PK_1, PK_2)$ as input. This is a reasonable alteration since PK is publicly available. Now, if $i \notin \mathcal{S}$, output $\perp$. Otherwise, use the result from Claim 7.2 to set

$$a_{\mathcal{S}} = \left(\prod_{v \in \mathcal{S}, v \neq i} Y_v^{t'}\right)$$

Let $\mathcal{C} = (c_0, \cdots, c_{m+1}, c_{m+2})$. Output the decrypted message as

$$\hat{\mathcal{M}} = c_{m+2} \frac{e(K_{\mathcal{S}}, U.c_0)}{e(c_{m+1}.a_{\mathcal{S}}, PK_2)}$$

Correctness. Refer Appendix D.1.

7.2 Tradeoffs

We point out certain advantages and disadvantages that this scheme has as compared to the previous constructions.

- No high-degree terms are generated hence the system administrator does not need the secret parameters for the multilinear map to setup the system. Randomization parameters are needed only for groups $\mathbb{G}_1$ and $\mathbb{G}_m$.

- The total multilinearity in handling 2^λ identities is $\lambda + 1$, as compared to 2λ and $1.44(\lambda + (\log_2 \lambda)/2) - 1$ respectively in the previous constructions.

- A flip side is that the ciphertext size is $O(\log N)$ group elements, as compared to $O(1)$ group elements in the previous constructions.

- The security proof for this scheme is in the generic multilinear map model, based on a better generic security theorem that for the previous constructions.

Finally, this construction can be generalized for multi-data owner systems just as in the previous constructions.

8 Conclusions

To be Written

References

[ACP09] 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.

[ADSFM12] 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.

[AFGH06] 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.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456. Springer, 2005.

[BCHL09] 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.

[BF03] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[BGW05] 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.

[BWZ14a] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In *Advances in Cryptology–CRYPTO 2014*, pages 206–223. Springer, 2014.

[BWZ14b] Dan Boneh, David J Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.

[CCT+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.

[CHK04] 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.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology–EUROCRYPT 2015*, pages 3–12. Springer, 2015.

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

[CLT13] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology–CRYPTO 2013*, pages 476–493. Springer, 2013.

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2014:975, 2014.

[CS02] 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.

[DMMM+12] Idilio Drago, Marco Mellia, Maurizio M Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. Inside dropbox: understanding personal cloud storage services. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 481–494. ACM, 2012.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Eurocrypt*, volume 7881, pages 1–17. Springer, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography*, pages 498–527. Springer, 2015.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. *IACR Cryptology ePrint Archive*, 2014:929, 2014.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[Mos10] Dana Moshkovitz. An alternative proof of the schwartz-zippel lemma. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 17, page 34, 2010.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

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

[PSM15] Sikhar Patranabis, Yash Shrivastava, and Debdeep Mukhopadhyay. Dynamic key-aggregate cryptosystem on elliptic curves for online data sharing. In *Progress in Cryptology–INDOCRYPT 2015*, pages 25–44. Springer, 2015.

A Relaxing Assumptions 3.5 And 4.5

In the two KAC constructions presented so far, we have assumed that all plaintext messages $\mathcal{M}$ may be efficiently embedded as elements in the respective target multilinear groups. However, embedding certain classes of plaintext data (such as multimedia) as group elements is extremely challenging and requires public samplability of group elements, which is insecure. However, a workaround may be readily proposed. Rather than embedding a random plaintext message $\mathcal{M}$ as a multilinear group element, we propose hashing the secret group element used to encrypt the message $\mathcal{M}$ itself using a collision-resistant hash function, and then combining it appropriately with the message $\mathcal{M}$. In order to ensure that the constructions are still provably secure in the standard model, we propose using *smooth projective hash functions* [CS02], that do not require the use of random oracles to prove security. Projective hash functions are very efficient to construct and can be designed to be collision-resistant [ACP09], making them an ideal choice for our constructions.

B CCA Secure KAC Using Asymmetric Multilinear Maps: Proofs

In this section, we state the proof of correctness for the CCA-secure KAC construction in Section 3.4, and also prove the claims 3.7 and 3.9.

B.1 Correctness

We show that the CCA-secure construction is correct, that is, $\hat{\mathcal{M}} = \mathcal{M}$ as follows.

$$
\begin{aligned}
\hat{\mathcal{M}} &= c_3 \frac{e(\hat{d}_1, U.c_1)}{e(\hat{d}_2, c_2)} \\
&= c_3 . \left(\frac{e\left(K_{\mathcal{S}}.SIG_{\mathcal{S}}.a_{\mathcal{S}}, g_{\mathbf{m}}^{t'}\right)}{e\left(b_{\mathcal{S}}, (PK.Y_i.Y_{2^m-1}^{V_{SIG}})^{t'}\right)} \right) . \left(\frac{e\left(\left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^w, g_{\mathbf{m}}^{t'}\right)}{e\left(g_{\mathbf{m}}^w, (PK.Y_i.Y_{2^m-1}^{V_{SIG}})^{t'}\right)} \right) \\
&= c_3 . \left(\frac{e\left(K_{\mathcal{S}}, g_{\mathbf{m}}^{t'}\right)}{e\left(b_{\mathcal{S}}, PK^{t'}\right)} \right) . \left(\frac{e\left(SIG_{\mathcal{S}}, g_{\mathbf{m}}^{t'}\right)}{e\left(b_{\mathcal{S}}, \left(Y_{2^m-1}^{V_{SIG}}\right)^{t'}\right)} \right) . \left(\frac{e\left(a_{\mathcal{S}}, g_{\mathbf{m}}^{t'}\right)}{e\left(b_{\mathcal{S}}, Y_i^{t'}\right)} \right) \\
&= c_3 \frac{e(\prod_{v \in \mathcal{S}, v \neq i} Y_{2^m-v+i}, g_{\mathbf{m}}^{t'})}{e(\prod_{v \in \mathcal{S}} Y_{2^m-v}, Y_i^{t'})} \\
&= \frac{\mathcal{M}.g_{\mathbf{2m}}^{t'\alpha^{(2^m)}}}{e(Y_{2^m}, g_{\mathbf{m}}^{t'})} \\
&= \mathcal{M}
\end{aligned}
$$

B.2 Proof of Claim 3.7

Recall that we have set

$$
\begin{aligned}
\hat{d}_1 &= (K_{\mathcal{S}}.SIG_{\mathcal{S}}.a_{\mathcal{S}}.(PK.Y_i.Y_{2^m-1}^{V_{SIG}})^w) \\
\hat{d}_2 &= (b_{\mathcal{S}}.g_{\mathbf{m}}^w)
\end{aligned}
$$

where w is uniformly randomly chosen from $\mathbb{Z}_q$. Set $x = w + \left(\sum_{v \in \mathcal{S}} \alpha^{2^m - v}\right)$. Quite evidently x is also uniformly random in $\mathbb{Z}_q$. Now, we have

$$\begin{aligned}
\hat{d}_2 &= (b_{\mathcal{S}}.g_{\mathbf{m}}^{w}) \\
&= g_{\mathbf{m}}^{\left(w + \left(\sum_{v \in \mathcal{S}} \alpha^{2^m - v}\right)\right)} \\
&= (g_{\mathbf{m}})^{x}
\end{aligned}$$

Also, we have the following:

$$\begin{aligned}
\hat{d}_1 &= (K_{\mathcal{S}}.SIG_{\mathcal{S}}.a_{\mathcal{S}}) . \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^{w} \\
&= (Y_{2^m})^{-1} (K_{\mathcal{S}}.SIG_{\mathcal{S}}.a_{\mathcal{S}}.Y_{2^m}) . \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^{w} \\
&= (Y_{2^m})^{-1} \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^{\left(\sum_{v \in \mathcal{S}} \alpha^{2^m - v}\right)} . \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^{w} \\
&= (Y_{2^m})^{-1} \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^{\left(w + \left(\sum_{v \in \mathcal{S}} \alpha^{2^m - v}\right)\right)} \\
&= (Y_{2^m})^{-1} \left(PK.Y_i.Y_{2^m-1}^{V_{SIG}}\right)^{x}
\end{aligned}$$

This concludes the proof of Claim 3.7. $\square$

B.3 Proof of Claim 3.9

Recall that we have set

$$\begin{aligned}
\hat{d}_0 &= Y_{2^m-1}^{(V_{SIG} - V_{SIG}^*)}.Y_v.Y_i^{-1} \\
\hat{d}_0' &= (Y_{v+1}/Y_{i+1})^{\frac{1}{(V_{SIG} - V_{SIG}^*)}} \\
\hat{d}_2 &= g_{\mathbf{m}}^{x}.Y_1^{\frac{1}{(V_{SIG} - V_{SIG}^*)}} \\
\hat{d}_1 &= \left(\hat{d}_2\right)^{u} . \left(\hat{d}_0\right)^{x} . \left(\hat{d}_0'\right)
\end{aligned}$$

where x is chosen at random from $\mathbb{Z}_q$. Also recall that $\mathcal{B}$ has set PK as $(g_{\mathbf{m}}^{u}) / \left(Y_i.Y_{2^m-1}^{V_{SIG}^*}\right)$, where u is chosen uniformly at random from $\mathbb{Z}_q$. Set $x' = x + \frac{\alpha}{(V_{SIG} - V_{SIG}^*)}$. Since x is uniform in $\mathbb{Z}_q$, so is x'. Now, we have

$$\begin{aligned}
\hat{d}_2 &= g_{\mathbf{m}}^{x}.Y_1^{\frac{1}{(V_{SIG} - V_{SIG}^*)}} \\
&= g_{\mathbf{m}}^{x}.g_{\mathbf{m}}^{\frac{\alpha}{(V_{SIG} - V_{SIG}^*)}} \\
&= g_{\mathbf{m}}^{x'}
\end{aligned}$$

Next, we have the following:

$$
\begin{aligned}
\hat{d}_1 &= \left(\hat{d}_2\right)^u . \left(\hat{d}_0\right)^x . \left(\hat{d}'_0\right) \\
&= (g^u_{\mathbf{m}})^{x'} . \left(Y_{2^m-1}^{x(V_{SIG}-V^*_{SIG})}\right) . (Y_v/Y_i)^{\left(x+\frac{\alpha}{(V_{SIG}-V^*_{SIG})}\right)} \\
&= \left(PK.Y_i.Y_{2^m-1}^{V^*_{SIG}}\right)^{x'} . \left(Y_{2^m-1}^{x(V_{SIG}-V^*_{SIG})}\right) . (Y_v/Y_i)^{x'} \\
&= \left(PK.Y_v.Y_{2^m-1}^{V_{SIG}}\right)^{x'} . \left(Y_{2^m-1}^{\left(x-x'\right)(V_{SIG}-V^*_{SIG})}\right) \\
&= \left(PK.Y_v.Y_{2^m-1}^{V_{SIG}}\right)^{x'} . \left(Y_{2^m-1}^{-\alpha}\right) \\
&= (Y_{2^m})^{-1} . \left(PK.Y_v.Y_{2^m-1}^{V_{SIG}}\right)^{x'}
\end{aligned}
$$

This concludes the proof of Claim 3.9. □

C KAC Using Symmetric Multilinear Maps: Proofs And Advanced Results

C.1 Correctness Of The CPA Secure Construction

Correctness may be established as follows.

$$
\begin{aligned}
\hat{\mathcal{M}} &= c_2 \frac{e(K_{\mathcal{S}}.a_{\mathcal{S}}, U.c_0)}{e(b_{\mathcal{S}}, c_1)} \\
&= c_2 \frac{e(\prod_{v\in\mathcal{S}} (Y_{2^m-1-v})^{\gamma} . \prod_{v\in\mathcal{S}, v\neq i} Y_{2^m-1-v+i}, g_l^{t'})}{e(\prod_{v\in\mathcal{S}} Y_{2^m-1-v}, (PK.\hat{Y}_i)^{t'})} \\
&= c_2 \frac{e(\prod_{v\in\mathcal{S}, v\neq i} Y_{2^m-1-v+i}, g_l^{t'})}{e(\prod_{v\in\mathcal{S}} Y_{2^m-1-v}, (\hat{Y}_i)^{t'})} \\
&= \frac{\mathcal{M}.g_{m+l-1}^{t'\alpha^{(2^m-1)}}}{e(Y_{2^m-1}, g_l^{t'})} \\
&= \mathcal{M}
\end{aligned}
$$

C.2 A CCA Secure Construction

We now demonstrate how to extend the identity-based KAC construction using multilinear maps to obtain non-adaptive chosen ciphertext security. We again make use of the signature scheme coupled with the collusion resistant hash function that we introduced in Section 3.4. In this CCA secure construction, we force the class index value i to be strictly less than $2^m - 2$ instead of $2^m - 1$ in the CPA secure construction.

SetUp$(1^\lambda, m)$: Same as in Section 4.1.

KeyGen$()$: Same as in Section 4.1.

Encrypt$(PK, i, \mathcal{M})$: Run the *SigKeyGen* algorithm to obtain a signature signing key K_{SIG} and a verification key $V_{SIG} \in \mathbb{Z}_q$. Randomly choose $r \in \mathbb{Z}_q$ and let $t' = t + r \in \mathbb{Z}_q$. Compute

$$
\mathcal{C}' = (g_l^r, (PK.Y_i.g_{m-1}^{V_{SIG}})^{t'}, \mathcal{M}.g_{m+l-1}^{t'\alpha^{(2^m-1)}})
$$

and output the ciphertext as

$$\mathcal{C} = (\mathcal{C}', Sign(\mathcal{C}', K_{SIG}), V_{SIG})$$

Extract$(msk, \mathcal{S})$: Same as in Section 4.1.

Decrypt$(\mathcal{C}, i, \mathcal{S}, K_{\mathcal{S}}, U)$: Let $\mathcal{C} = ((c_0, c_1, c_2), \sigma, V_{SIG})$. Verify that σ is a valid signature of (c_0, c_1, c_2) under the key V_{SIG}. If not, output $\perp$. Also, if $i \notin \mathcal{S}$, output $\perp$. Otherwise, set

$$SIG_{\mathcal{S}} = \prod_{v \in \mathcal{S}} g_{m-1}^{V_{SIG}}$$
$$a_{\mathcal{S}} = \prod_{v \in \mathcal{S}, v \neq i} Y_{2^m-1-v+i}$$
$$b_{\mathcal{S}} = \prod_{v \in \mathcal{S}} Y_{2^m-1-v}$$

Next, pick a random $w \in \mathbb{Z}_q$ and set

$$\hat{d}_1 = (K_{\mathcal{S}}.SIG_{\mathcal{S}}.a_{\mathcal{S}}.(PK.Y_i.g_{m-1}^{V_{SIG}})^w)$$
$$\hat{d}_2 = (b_{\mathcal{S}}.g_{m-1}^{w})$$

Output the decrypted message

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

The proof of correctness of this scheme is presented in Appendix B.1. Note that the overhead for the ciphertext, aggregate key, public parameters, and the private and public keys, remains unchanged. The main change from the original scheme is in the fact that decryption requires a randomization value $w \in \mathbb{Z}_q$.

Claim C.1. *For a given $i \in \mathcal{S}$, the pair $(\hat{d}_1, \hat{d}_2)$ is chosen from the following distribution*

$$\left((Y_{2^m-1})^{-1} . \left(PK.Y_i.g_{m-1}^{V_{SIG}}\right)^x, (g_{m-1})^x\right)$$

where x is uniformly randomly chosen from $\mathbb{Z}_q$.
Proof. Similar to the proof in Appendix B.2.

Once again, note that the distribution $(\hat{d}_1, \hat{d}_2)$ depends *only on the data class i for the message $\mathcal{M}$ to be decrypted* and is *completely independent of the subset $\mathcal{S}$ used to encrypt it.* We next prove the non-adaptive CCA security of this scheme.

Theorem C.2. *Let **Setup**$'$ be the setup algorithm for a symmetric multilinear map, and let the decisional (m, l)-Multilinear Diffie-Hellman Exponent assumption holds for SetUp$'$. Then our proposed construction of KAC for N' data classes presented in this section is non-adaptively CCA secure for $N' = \binom{m}{l}$, where each data class number $i < 2^m - 2$.*

Proof. Let $\mathcal{A}$ be a poly-time adversary such that $|Adv_{\mathcal{A},N'} - \frac{1}{2}| > \epsilon$ for the proposed KAC system parameterized with an identity space $\mathcal{ID}'$ of size N', and ϵ is a non-negligible positive constant. We build an algorithm $\mathcal{B}$ that has advantage at least ϵ in solving the decisional (m, l)-MDHE problem for **Setup**$'$. $\mathcal{B}$ takes as input a random (m, l)-MDHE challenge tuple $(params', \{X_j\}_{j \in \{0,\cdots,m\}}, V, Z)$, where:

- $param' \leftarrow SetUp'(1^\lambda, m+l-1)$
- $X_j = g_1^{\alpha^{(2^j)}}$ for $0 \le j \le m$
- $V = g_l^{t'}$ for a random $t' \in \mathbb{Z}_q$ (where q is a λ bit prime)
- Z is either $g_{m+l-1}^{t'\alpha^{(2^m-1)}}$ or a random element of $\mathbb{G}_{m+l-1}$.

$\mathcal{B}$ then proceeds as follows.

Commit: $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S}^*$ of data classes that $\mathcal{A}$ wishes to be challenged on. $\mathcal{B}$ then randomly chooses a data class $i \in \mathcal{S}^*$ and provides it to $\mathcal{A}$.

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 $V_{SIG}^* \in \mathbb{Z}_q$. The various items to be provided to $\mathcal{A}$ are generated as follows.

- $param$ is set as $(param'', \{X_i\}_{i\in\{0,\cdots,m\}})$.
- PK is set as $(g_l^u)/\big(Y_i.gV_{SIG}^*{}_{m-1}\big)$ where u is chosen uniformly at random from $\mathbb{Z}_q$. Note that this is equivalent to setting $msk = u - \alpha^i - V_{SIG}^*$, as required.
- U is set as $g_{\mathbf{m}}^t$ where t is again chosen uniformly at random from $\mathbb{Z}_q$.
- $\mathcal{B}$ then computes

  $$K_{\overline{\mathcal{S}^*}} = \prod_{v \notin \mathcal{S}^*} \frac{Y_{2^m-1-v}^u}{(Y_{2^m-1-v+i}).(gV_{SIG}^*{}_{m-1})}$$

  Observe that $K_{\overline{\mathcal{S}^*}} = \prod_{v\notin\mathcal{S}^*} PK^{\alpha^{2^m-1-v}}$, as desired. Moreover, $\mathcal{B}$ is aware that $i \notin \overline{\mathcal{S}^*}$ (implying $i \neq v$), and hence has all the resources to compute $K_{\overline{\mathcal{S}^*}}$.

Since the $g_{\mathbf{m}}$, α, u, t' and t values are chosen uniformly at random, *the public parameters and the public, private and authentication keys have an identical distribution to that in the actual construction.*

Query Phase 1: Algorithm $\mathcal{A}$ now issues decryption queries. Let $(\mathcal{C}, v)$ be a decryption query $\mathcal{C}$ is obtained by $\mathcal{A}$ using some subset $\mathcal{S}$ containing v. However, $\mathcal{B}$ is not given the knowledge of $\mathcal{S}$. 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

$$\hat{d}_0 = Y^{(V_{SIG}-V_{SIG}^*)}.Y_v.Y_i^{-1}$$
$$\hat{d}_0' = (Y_{v+1}/Y_{i+1})^{\overline{\frac{1}{(V_{SIG}-V_{SIG}^*)}}}$$
$$\hat{d}_2 = g_{\mathbf{m}}^x.Y_1^{\overline{\frac{1}{(V_{SIG}-V_{SIG}^*)}}}$$
$$\hat{d}_1 = \left(\hat{d}_2\right)^u.\left(\hat{d}_0\right)^x.\left(\hat{d}_0'\right)$$

Note that $\hat{d}_0'$ can be computed following Claim 4.1.b as $1 \le i, v \le 2^m - 3$. $\mathcal{B}$ responds with $K = c_2\frac{e(\hat{d}_1, c_0.U)}{e(\hat{d}_2, c_1)}$.

Claim C.3. *$\mathcal{B}$'s response is exactly as in a real attack scenario, that is, for some x' chosen uniformly at random from $\mathbb{Z}_q$, we have*

$$\hat{d}_1 = (Y_{2^m-1})^{-1}.\left(PK.Y_v.g_{m-1}^{V_{SIG}}\right)^{x'} \text{ and } \hat{d}_2 = g_{m-1}^{x'}$$

Proof. Similar to the proof in Appendix B.3.

Thus, by the result in Claim C.1, $\mathcal{B}$'s response is identical to **Decrypt**$(\mathcal{C}, v, \mathcal{S}, K_{\mathcal{S}}, U)$, even though $\mathcal{B}$ does not possess the knowledge of the subset $\mathcal{S}$ used by $\mathcal{A}$ to obtain $\mathcal{C}$.

Challenge: $\mathcal{A}$ picks at random two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ from the set of possible plaintext messages in $\mathbb{G}_{\mathbf{2m}}$, and provides them to $\mathcal{B}$. $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and sets

$$\mathcal{C} = (U^{-1}V, V^u, \mathcal{M}_b.Z)$$
$$\mathcal{C}^* = (\mathcal{C}, Sign(\mathcal{C}, K^*_{SIG}), V^*_{SIG})$$

The challenge posed to $\mathcal{A}$ is $(\mathcal{C}^*, \mathcal{M}_0, \mathcal{M}_1)$. It can be easily shown that when $Z = g_{m+l-1}^{t'\alpha^{(2^m-1)}}$ (i.e. the input to $\mathcal{B}$ is a valid m-MDHE tuple), then this 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 (indicating that $Z = g_{m+l-1}^{t'\alpha^{(2^m-1)}}$). Otherwise, it outputs 1 (indicating that Z is a random element in $\mathbb{G}_{m+l-1}$).

It can be easily proved that the probability that $\mathcal{B}$ aborts the simulation as a result of one of the decryption queries by $\mathcal{A}$ is less than ϵ_2 (from the existential unforgability property of the signature scheme). We conclude that $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$, which must therefore be negligible, as desired. This completes the proof of Theorem C.2. $\square$

D The Third KAC Construction: Proofs

D.1 Correctness

Correctness of the third KAC construction may be established as follows.

$$\begin{aligned}
\hat{\mathcal{M}} &= c_{m+2} \frac{e(K_{\mathcal{S}}, U.c_0)}{e(c_{m+1}.a_{\mathcal{S}}, PK_2)} \\
&= c_{m+2} \frac{e\left(\left(\prod_{v \in \mathcal{S}} Y_v\right)^x, g_1^{t'}\right)}{e\left(\left(g_m^{\gamma}.Y_i.\prod_{v \in \mathcal{S}, v \neq i} Y_v\right)^{t'}, g_1^x\right)} \\
&= \frac{c_{m+2}}{e((g_m^{\gamma})^{t'}, g_1^x)} \\
&= \mathcal{M}
\end{aligned}$$

D.2 Security In The Generic Multilinear Map Model

In this section, we prove that our third KAC construction is adaptively CPA secure in the generic multilinear map model described in Section 6. In particular, we demonstrate that with a prime group order parameter $q \approx 2^{\lambda}$, the scheme achieves λ-bit security. We state and prove the following theorem.

Theorem D.1. *Any generic adversary $\mathcal{A}$ that can make at most a polynomial number of queries to (**Encode**,**Mult**,**Pair**) has negligible advantage in breaking the adaptive security of the KAC construction presented in Section 7.1, provided that $1/q$ is negligible.*

Proof. Let $\mathcal{A}$ be an adaptive adversary under the generic model and let $\mathcal{B}$ be a challenger that plays the following game with $\mathcal{A}$:

SetUp: Challenger $\mathcal{B}$ sets up the system for $\mathcal{ID}$ consisting of all m bit class identities. $\mathcal{B}$ generates random $\alpha_{j,b} \in \mathbb{Z}_q$ for $j = 0, \cdots, m-1$, and $b = 0, 1$. $\mathcal{B}$ also generates random $\gamma, x, t \in \mathbb{Z}_q$. $\mathcal{A}$ receives the following:

- $X_{j,b} = \xi(\alpha_{j,b}, 1)$ for $j = 0, \cdots, m-1$, and $b = 0, 1$
- $PK = (\xi(\gamma, m), \xi(x, 1))$
- $U = \xi(t, 1)$

Oracle Query Phase. $\mathcal{A}$ adaptively issues queries to (**Encode**,**Mult**,**Pair**).

Commit. Algorithm $\mathcal{A}$ commits to a set $\mathcal{S} \subset \mathcal{ID}$ of data classes that it wishes to attack. Since collusion attacks are allowed in our framework, $\mathcal{B}$ furnishes $\mathcal{A}$ with the aggregate key $K_{\overline{\mathcal{S}}}$ computed as

$$K_{\mathcal{S}} = \xi\left(msk_2 \sum_{v \notin \mathcal{S}} \prod_{j=0}^{m-1} \alpha_{j,v_j}, m\right)$$

Challenge: To make a challenge query, $\mathcal{A}$ randomly generates $\hat{m}_0$ and $\hat{m}_1$ and provide these to $\mathcal{B}$. $\mathcal{B}$ computes $\mathcal{M}_0 = \xi(\hat{m}_0, m+1)$ and $\mathcal{M}_1 = \xi(\hat{m}_1, m+1)$ and randomly chooses a class $i \in \mathcal{S}$. $\mathcal{B}$ also randomly chooses $r \in \mathbb{Z}_q$ and sets $t' = t + r \in \mathbb{Z}_q$. Then it sets

$$\begin{aligned} c_0 &= \xi(r, 1) \\ c_{j+1} &= \xi(t'\alpha_{j,(1-u_j)}) \text{ for } j = 0, 1, \cdots, m-1 \\ c_{m+1} &= \xi(t'(\gamma + \prod_{j=0}^{m-1} \alpha_{j,i_j}), m) \\ c_{m+2} &= \mathcal{M}_{\hat{b}} + \xi(\gamma x t', m+1) \end{aligned}$$

Finally, $\mathcal{B}$ sets

$$\mathcal{C} = (\{c_j\}_{j \in \{0, \cdots, m+2\}})$$

and provides the challenge to $\mathcal{A}$ as $(\mathcal{C}, \mathcal{M}_0, \mathcal{M}_1)$.

Guess: $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, $\mathcal{A}$ wins the game.

We now assume that the algorithm $\mathcal{B}$ maintains a list $L = \{(p_k, j_k, \epsilon_k)\}$, where p_k is a polynomial in the formal variables $(\{\alpha_{j,b}\}_{j \in \{0, \cdots, m-1\}, b \in \{0,1\}}, \gamma, x, t, r, \hat{m}_0, \hat{m}_1, \mathcal{M}_{\hat{b}})$, j_k indexes the groups and $\epsilon_k \in \{0, 1\}^n$. The list is initialized with the following tuples:

- $(\alpha_{j,b}, 1, \xi_{2j+b})$ for randomly generated strings $\xi_{2j+b} \in \{0, 1\}^n$ for some n, $j \in \{0, \cdots, m-1\}$, $b \in \{0, 1\}$
- $(\gamma, m, \xi 2m)$ for a randomly generated $\xi_{2m} \in \{0, 1\}^n$
- (x, m, ξ_{2m+1}) for a randomly generated $\xi_{2m+1} \in \{0, 1\}^n$
- (t, m, ξ_{2m+2}) for a randomly generated $\xi_{2m+2} \in \{0, 1\}^n$

Thus intially, $|L| = 2m+3$ and the game begins with $\mathcal{B}$ supplying the set of strings $\{\xi_j\}_{j \in \{0, \cdots, 2m+2\}}$ to $\mathcal{A}$, who can make some queries as described next.

Table 1: Upper Bounds on Contributions to Length of L

Query Stage	**Maximum Contribution to** $|L|$
SetUp	$2m+3$
Oracle Query Phase	$Q_e + Q_m + Q_p$
Commit	1
Challenge	$m+5$
Total	$Q_e + Q_m + Q_p + 3m + 9$

- $\mathcal{A}$ is allowed at most a polynomial number of queries to (**Encode**,**Mult**,**Pair**). We avoid presenting the details of how $\mathcal{B}$ handles these queries. Refer [BWZ14a] for details. We point out, however, that in each query to any of the three algorithms, at most one new tuple is added to L, and no tuple can have index $i_j > m+1$.

- $\mathcal{A}$ is allowed to commit to a set $\mathcal{S}$ and query for the collusion key. $\mathcal{B}$ adds the tuple $(\xi\left(x\sum_{v\notin\mathcal{S}}\prod_{j=0}^{m-1}\alpha_{j,v_j}, m\right), n, \xi)$ randomly generated $\xi \in \{0,1\}^n$, which is given as response to $\mathcal{A}$.

- Finally, $\mathcal{A}$ is allowed to make a single challenge query on a class $i \in \mathcal{S}$. $\mathcal{B}$ creates new formal variables $(r, \mathcal{M}_{\hat{b}}, \hat{m}_0, \hat{m}_1)$ and adds the following tuples to its list L:

  - $(r, 1, \xi)$
  - $((t+r)\alpha_{j,(1-i_j)}, 1, \xi_j)$ for $j = 0, 1, \cdots, m-1$
  - $((t+r)(\gamma + \prod_{j=0}^{m-1}\alpha_{j,i_j}), m, \xi_m)$
  - $(\mathcal{M}_{\hat{b}} + \xi(\gamma x t', m+1, \xi_{m+1})$
  - $(\hat{m}_0, 1, \xi_{m+2})$
  - $(\hat{m}_1, 2, \xi_{m+3})$

  Once again ξ and $\xi_j, j \in \{0, \cdots, m+3\}$ are randomly generated strings in $\{0,1\}^n$ that are provided to $\mathcal{A}$ as response.

- $\mathcal{A}$ outputs a guess $\hat{b}' \in \{0,1\}$.

Now, $\mathcal{B}$ chooses random values for $\alpha_{j,b}, \gamma, x, t, r, \hat{b}'$. $\mathcal{A}$ picks at random $\hat{m}_0$ and $\hat{m}_1$, and provides them to $\mathcal{B}$.

It can be easily seen that $\mathcal{B}$ simulates $\mathcal{A}$ perfectly unless the choices for the parameters $\alpha_{j,b}, \gamma, x, t, r, \hat{b}$ does not result in a *false polynomial equality event* Υ (where $j_{k_1} = j_{k_2}$ but $p_{k_1} \neq p_{k_2}$ for random $k_1 \neq k_2$). We next make the following observations.

Observation D.2. *The maximum degree of any polynomial in the above simulation is at most* $2m+2$.

Observation D.3. $\mathcal{A}$ *cannot trigger a false polynomial equality event* Υ *for any choice of* $\hat{m}_0, \hat{m}_1$.

We now look at the probability that a random choice for the parameters $\alpha_{j,b}, \gamma, x, t, r, \hat{b}$ results in a *false polynomial equality event* Υ. We first state and prove the following claim.

Claim D.4. For $p_{k_1} \neq p_{k_2}$ the probability that $p_{k_1}(\alpha_{j,b}, \cdots) = p_{k_2}(\alpha_{j,b}, \cdots)$ is upper bounded by $(2m+2)/q$, where q is the prime group order parameter.

Proof. By Observation D.2. and the Swartz-Zippel lemma [Mos10].

Assume that $\mathcal{A}$ makes Q_E queries to **Encode**, Q_M queries to **Mult** and Q_P queries to **Pair** during **Oracle Query Phase**. Table 1 summarizes the maximum possible contributions to $|L|$ by the tuples added by $\mathcal{B}$ at different query stages. Note that $|L|$ is upper bounded by $(Q_e + Q_m + Q_p + 3m + 9)$. From Observation D.3. and Claim D.4., it easily follows that the probability of a false polynomial equality event Υ is upper bounded as

$$Pr(\Upsilon) \leq \left(Q_e + Q_m + Q_p + 3m + 9\right)^2 (m+2)/2q$$

If Υ does not occur, $\mathcal{B}$ simulates $\mathcal{A}$ perfectly and, from $\mathcal{A}$'s view, $\hat{b}$ is independent as it was chosen after the simulation. So, it is straightforward to conclude that

$$|Adv_{\mathcal{A},2^m} - \frac{1}{2}| \leq \left(Q_e + Q_m + Q_p + 3m + 9\right)^2 (m+1)/2q$$

For polynomial Q_e, Q_m, Q_p, m this quantity is negligible provided that $1/q$ is negligible, as desired. This completes the proof of Theorem D.1. □