IDENTITY-BASED KEY AGGREGATE CRYPTOSYSTEM FROM MULTILINEAR MAPS

Sikhar Patranabis and Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
West Bengal 721302, India

Abstract. A key-aggregate cryptosystem (KAC) is the dual of the well-known notion of broadcast encryption (BE). In KAC, each plaintext message is encrypted with respect to some identity, and a single aggregate key can be generated for any arbitrary subset $S$ of identities, such that any ciphertext designated for any identity in $S$ can be decrypted using this aggregate key. A KAC scheme is said to be efficient if all public parameters, ciphertexts and aggregate keys have polynomial overhead, and can be generated using poly-time algorithms.

A KAC scheme is said to be identity-based if remains efficient even when the number of unique identities supported by the system is exponential in the security parameter $\lambda$. Unfortunately, existing KAC constructions do not satisfy this property. In particular, adopting these constructions to the identity-based setting leads to public parameters with exponential overhead.

In this paper, we propose new identity-based KAC constructions using multilinear maps that are secure in the generic multilinear map model, and are fully collusion resistant against any number of colluding parties. Our first construction is based on asymmetric multilinear maps, with a poly-logarithmic overhead for the public parameters, and a constant overhead for the ciphertexts and aggregate keys. Our second construction is based on the more generalized symmetric multilinear maps, and offers tighter security bounds in the generic multilinear map model. This construction has a poly-logarithmic overhead for the public parameters and the ciphertexts, while the overhead for the aggregate keys is still constant.

1. Introduction

A key-aggregate cryptosystem (KAC), introduced by Chu et al. in [7] and studied subsequently in [17, 18], is the dual of the well-known cryptographic notion of broadcast encryption (BE) [2]. KAC schemes have a number of different real-life applications, most notably in the context of selectively sharing encrypted data among multiple recipients in a cloud-based environment.

In BE, a broadcaster encrypts a single message for some subset $S$ of users who are listening on a broadcast channel. Each user is associated with a unique identity, and can use his private key to decrypt the broadcast ciphertext. However, even if all users outside of $S$ collude, they can obtain no information about the contents of the broadcast. Such a BE scheme is said to be collusion resistant.

In KAC, each plaintext message is encrypted with respect to an individual identity, and a single aggregate key can be generated for any arbitrary subset $S$ of

2010 Mathematics Subject Classification: Primary: 11T71, 68P25, 94A60; Secondary: 68R01.

Key words and phrases: Key-aggregate, identity-based, multilinear maps, collusion-resistant, standard model, generic group model.
identities, such that any ciphertext designated for any identity in $S$ can be decrypted using this aggregate key. The notion of collusion-resistance in the context of KAC is defined as follows: consider any collection of decryptors that have access to decryption keys for all identities outside $S$, but not for any identity in $S$; a KAC scheme is said to be collusion-resistant if such a collection of decryptors can obtain no information about the message underlying any ciphertext designated for any identity in $S$.

**Identity-Based KAC.** A KAC scheme is said to be efficient if all public parameters, ciphertexts and aggregate keys have polynomial overhead, and can be generated using poly-time algorithms. A KAC scheme is said to be identity-based if remains efficient even when the number $N$ of unique identities is exponential in the security parameter $\lambda$.

Unfortunately, existing KAC constructions [7, 17, 18] do not satisfy this property. In particular, these constructions have public parameters with overhead linear in the number of supported identities $N$. In other words, adopting these constructions to the identity-based setting leads to public parameters with exponential overhead. This motivates us to design new constructions for KAC that are identity-based.

**Our Contributions.** In this paper, we propose two new identity-based KAC constructions using multilinear maps with at most poly-logarithmic overhead (in the number of identities $N$) for all system parameters, ciphertexts and aggregate keys.

1. Our first KAC construction uses an asymmetric $O(\log N)$ multilinear map to support $N = 2^m - 1$ identities. The scheme has a public parameter overhead of $O(\log N)$ group elements, and produces short ciphertexts and aggregate keys comprising of $O(1)$ group elements. The scheme is adaptively secure in the generic multilinear group model and is fully collusion resistant against any number of colluding parties.

2. Our second KAC construction is based on the more generalized symmetric multilinear maps, and provides tighter bounds on the group size parameters for adaptive security in the generic multilinear map model as compared to the first construction. There exists a trade-off in the form of a blow-up of the ciphertext size, which is $O(\log N)$ group elements instead of $O(1)$. The rest of the parameters have the same overhead as in the first construction. The second construction is again identity-based, since the overall system overhead still remains poly-logarithmic. In addition, the second construction is fully collusion resistant against any number of colluding parties.

Technically, our constructions are inspired by the identity-based BE schemes from multilinear maps proposed by Boneh, Waters and Zhandry in [4]. The original work of Chu et al. [7] presents a framework to extend a BE scheme supporting polynomially many identities (more specifically, the bilinear map-based BE scheme of Boneh, Gentry and Waters [2]) to a KAC scheme supporting polynomially many identities. In this paper, we follow a similar strategy to build identity-based KAC schemes.

2. Preliminaries

In this section, we recall the formal definition of a key-aggregate cryptosystem (KAC) from [7].
2.1. KEY-AGGREGATE CRYPTOSYSTEM: THE BASIC FRAMEWORK. A KAC scheme consists of four probabilistic polynomial-time algorithms that are described next:

- **SetUp**$(\lambda, \mathcal{ID})$: The setup algorithm takes as input the security parameter $\lambda$ and an identity space $\mathcal{ID}$, and outputs the public parameter $\text{param}$ and the master secret key $\text{msk}$.

- **Encrypt**$(\text{param}, i, \mathcal{M})$: Takes as input the public parameter $\text{param}$, an identity $i \in \mathcal{ID}$ and the plaintext message $\mathcal{M}$. Outputs the corresponding ciphertext $\mathcal{C}$.

- **Extract**$(\text{param}, \text{msk}, S)$: Takes as input the public parameter $\text{param}$, the master secret key $\text{msk}$ and a polynomial-size subset of identities $S \subseteq \mathcal{ID}$. Computes the aggregate decryption key $K_S$ for the subset $S$.

- **Decrypt**$(\text{param}, \mathcal{C}, K_S)$: Takes as input the public parameter $\text{param}$, a ciphertext $\mathcal{C}$ corresponding to an identity $i \in S$ and an aggregate key $K_S$. Outputs either a message $\mathcal{M}$ or $\perp$.

**Correctness.** A KAC scheme is said to be correct if for any valid identity space $\mathcal{ID}$, any set $S \subseteq \mathcal{ID}$, any identity $i \in S$, and any plaintext message $\mathcal{M}$, we must have

$$\Pr[\text{Decrypt}(\text{param}, \mathcal{C}, K_S) = \mathcal{M}|E] = 1$$

where $E$ is the event described as the conjunction of the following atomic events:

$$\text{(param, msk)} \leftarrow \text{SetUp}(\mathcal{ID}), \mathcal{C} \leftarrow \text{Encrypt}(\text{param}, i, \mathcal{M}), \quad K_S \leftarrow \text{Extract}(\text{param}, \text{msk}, S)$$

2.2. SECURITY DEFINITIONS. We now recall the 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 $\mathcal{ID}$ 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 $\text{param}$ and provides the same to $\mathcal{A}$.

**Aggregate Key Query Phase 1.** : Algorithm $\mathcal{A}$ adaptively issues aggregate key queries for any subset $\tilde{S} \subseteq \mathcal{ID}$. $\mathcal{B}$ responds with the aggregate key $K_{\tilde{S}}$.

**Decryption Query Phase 1.** : Algorithm $\mathcal{A}$ adaptively issues decryption queries $q_1, \cdots, q_w$. Here a decryption query comprises of the tuple $(\mathcal{C}, v)$, where $v \in \mathcal{ID}$ is the identity associated with $\mathcal{C}$. The challenger has to respond with a valid decryption of the ciphertext.

**Commit.** : $\mathcal{A}$ adaptively commits to an identity $i \in \mathcal{ID}$, subject to the constraint that $i \notin \tilde{S}$ for all subsets $\tilde{S}$ queried in the aggregate key query phase 1.

**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 $\mathcal{C}^* = \text{Encrypt}(\text{param}, i, \mathcal{M}_b)$.

**Aggregate Key Query Phase 2.** : Algorithm $\mathcal{A}$ continues to adaptively issue aggregate key queries for any subset $\tilde{S} \subseteq \mathcal{ID}$ such that $i \notin \tilde{S}$. $\mathcal{B}$ responds with the aggregate key $K_{\tilde{S}}$.

**Decryption Query Phase 2.** : $\mathcal{A}$ 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 is now subject to the restriction $\mathcal{C} \neq \mathcal{C}^*$. $\mathcal{B}$ responds as in decryption query phase 1.
Guess. : $A$ outputs a guess $b'$ of $b$. If $b' = b$, $A$ wins the game.

The game above models an attack in the real world setting where users who do not have authorized access to the subset $S$ collude to try and expose a message in this subset. We now formally define the security notions for KAC. Let $\text{Adv}_{A,|ID|}$ denote the probability that $A$ wins the game.

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

**Definition 2.2.** A KAC construction is $(\epsilon, ID)$ adaptively secure under a chosen plaintext attack (that is, adaptively CPA-secure) if it is $(\epsilon, ID, 0)$ 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 $A$ is required to commit to the identity $i$ 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 $(\epsilon, ID, Q_D)$ non-adaptively secure under a chosen ciphertext attack (that is, non-adaptively CCA-secure) if, for all non-adaptive probabilistic poly-time algorithms $A$ that can make a total of $Q_D$ decryption queries, we have that $|\text{Adv}_{A,|ID|} - \frac{1}{2}| < \epsilon$.

**Definition 2.4.** A KAC construction is $(\epsilon, ID)$ non-adaptively secure under a chosen plaintext attack (that is, non-adaptively CPA-secure) if it is $(\epsilon, ID, 0)$ 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 popular candidate multilinear map constructions [11].

2.3.1. **Symmetric multilinear maps.** A standard symmetric multilinear map consists of the following pair of algorithms.

$\text{SetUp}'(1^\lambda, m)$: Sets up an $m$-linear map by outputting an $m$-tuple of groups $< G_1, G_2, \cdots, G_m >$ of prime order $q$ (where $q$ is a $\lambda$ bit prime), along with the respective generator $g_i \in G_i$ for $1 \leq i \leq m$. In standard notation, $G_1$ is the source group, $G_m$ is the target group, and $G_2, \cdots, G_{m-1}$ are the intermediate groups.

$e_{i,j}(h_1, h_2)$: Takes as input $h_1 \in G_i$ and $h_2 \in G_j$, and outputs $h_3 \in G_{i+j}$ 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. Although symmetric multilinear maps are simple to understand, there currently exist no reported candidate constructions for the same. Almost all candidate constructions [11, 9] realize asymmetric multilinear maps, which we describe next.
2.3.2. Asymmetric multilinear maps. We adopt the same definition of asymmetric multilinear maps presented in [11]. 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.

\textbf{setUp}′′(1\^{\lambda}, \bf{n}):\ Takes as input a vector $\bf{n} \in \mathbb{Z}^l$. Sets up an $\bf{n}$-linear map by outputting an $n$-tuple of groups $<G_1, G_2, \ldots, G_n>$ of prime order $q$ (where $q$ is a $\lambda$ bit prime), along with the respective generators $g_v \in G_v$ for $1 \leq v \leq n$ (comparision is defined component-wise). Further, let $x_i$ be the $i$th standard basis vector (with 1 at position $i$ and 0 at each other position). In standard notation, $G_{x_i}$ is the $i$th source group, $G_v$ is the target group, and the rest are the intermediate groups.

$e_{v_1,v_2}(h_1, h_2)$: Takes as input $h_1 \in G_{v_1}$ and $h_2 \in G_{v_2}$, and outputs $h_3 \in G_{i+j}$ such that

$$(h_1 = g_{v_1}^a, h_2 = g_{v_2}^b) \Rightarrow h_3 = g_{i+j}^{ab}$$

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

2.4. SOME HARDNESS ASSUMPTIONS RELATED TO MULTILINEAR MAPS. In this section, we introduce two hardness assumptions that are relevant to our discussions.

\textbf{The Hybrid Diffie-Hellman Exponent Assumption.} Let param'' is generated by \textbf{setUp}'′(1\^{\lambda'}, 2\bf{m})$, where $\bf{m}$ is the $m + 1$ length vector consisting of all ones and $\lambda'$ is the adequate group size parameter for achieving $\lambda$ bit security. Choose $\alpha \in \mathbb{Z}_q$ at random (where $q$ is a $\lambda'$-bit prime), and let $X_j = g_{X_j}^{(2^j)}$ for $0 \leq j \leq m - 1$. Also, define $X_m = g_{X_m}^{2^{(2m+1)}}$. Choose a random $t \in \mathbb{Z}_q$, and let $V = g_{2m}^t$. The decisional $m$-Hybrid Diffie Hellman Exponent (HDHE) problem as defined as follows. Given the tuple $\{\text{param}''$, $\{X_j\}_{j \in \{0, \ldots, m\}}$, $V$, $Z\}$, distinguish if $Z$ is $g_{2m}^{2^j}$ or a random element of $G_{2^m}$.

2.5. MULTILINEAR MAP REQUIREMENTS. In this paper, 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 only do this to make the analysis simple and easy to follow. In general, the KAC constructions presented in this paper can be instantiated using multilinear maps with the properties listed below:

- The representation of an element should be statistically independent of the group and multilinear operations that led to that element.
- It is possible to extract a \textit{canonical} representation of an element in the target group given any representation of that element using the \textit{zero-test parameter}.
- The party setting up the multilinear map has sufficient \textit{trapdoor} information to compute $g^{\alpha x}$ for a non-random $\alpha$ and exponentially large $x$.
- It is possible to generate asymmetric multilinear maps for any positive integer vector $\bf{n} \in \mathbb{Z}^l$.
- 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.
The two foremost candidate multilinear map constructions [11, 9] possess the aforementioned properties. Unfortunately, both these constructions, as well as attempted fixes for them [13, 5] have been cryptanalyzed either in parts or completely [6, 10].

A more recent proposition by Gentry et al. is the graph-induced multilinear map based on non-ideal lattices [12], which naturally gives rise to asymmetric multilinear maps. Even this construction has been broken by Coron et al. [8], who demonstrate how to recover an equivalent private key for this scheme (even in the presence of safeguards).

2.6. THE GENERIC MULTILINEAR MAP MODEL. Given the extensive cryptanalysis [6, 10] of the existing multilinear map constructions [11, 9, 13, 5, 12], most of the intractable problems related to multilinear maps are currently secure only in the generic multilinear map model. Just as multilinear maps are an extension of bilinear maps, the generic multilinear map model is an extension of the generic bilinear map model [1]. We describe the generic multilinear map model here for completeness.

In the generic multilinear map model [4], the group \( \mathbb{G}_v \) (where \( v \in \mathbb{Z}_l \)) is represented by a random injective function \( \xi: \mathbb{Z}_q \times \mathbb{Z}_l \rightarrow \{0, 1\}^n \) that maps pairs of elements from the additive group \( \mathbb{Z}_q \) and the vector space \( \mathbb{Z}_l \) to random strings of sufficiently large length. The random string output by \( \xi(\alpha, v) \) for \( \alpha \in \mathbb{Z}_q \) and \( v \in \mathbb{Z}_l \) is thus essentially a randomized representation in the generic model, corresponding to the group element \( g^v \) in the standard multilinear map definition.

For example, suppose that the target vector in a generic multilinear map construction is denoted by \( m \in \mathbb{Z}_l \). Any algorithm in the generic multilinear map model is said to interact with the map using the tuple of algorithms (\( \text{Encode}, \text{Mult}, \text{Pair} \)) described below.

\begin{align*}
\text{Encode}(x, v) & : \text{Takes as input a non-negative integer vector } v \leq m \text{ and outputs } \xi(x, v). \\
\text{Mult}(\xi_1, \xi_2, \cdot) & : \text{Takes as input } \xi_1 = \xi(x_1, v), \; \xi_2 = \xi(x_2, v) \text{ and an operation } \cdot \in \{+, -\}. \text{ Outputs } \xi(x_1 \cdot x_2, v). \\
\text{Pair}(\xi_1, \xi_2) & : \text{Takes as input } \xi_1 = \xi(x_1, v_1) \text{ and } \xi_2 = \xi(x_2, v_2) \text{ where } v_1 + v_2 = v \leq m. \text{ Outputs } \xi(x_1, x_2, v).
\end{align*}

Note that if the inputs are not valid, each of the above algorithms returns \( \perp \). Also, \( \text{Mult} \) and \( \text{Pair} \) here are assumed to be oracles to compute the induced group multiplication and multilinear map operations. Finally, note that in this paper, we mostly consider binary vectors in the group \( \mathbb{Z}_l \), and hence choose to denote them by their corresponding integer representations for ease of representation.

3. KAC USING ASYMMETRIC MULTILINEAR MAPS

In this section, we present the first construction of identity-based KAC based on asymmetric multilinear maps. Our goal in this scheme is to support \( N = 2^m - 1 \) identities, where \( m \) is polynomially large in the security parameter \( \lambda \), while ensuring that the size of the public parameter is \( O(\log N) \) group elements, while the ciphertext and the aggregate key comprise of \( O(1) \) group elements.

3.1. THE BASIC IDEA. The core idea of our construction is inspired by techniques presented in [4]. Let \( N = 2^m - 1 \) for some integer \( m \), and let \( m \) be the \( m + 1 \)
length vector consisting of all ones. We use an asymmetric multilinear map with the target group $\mathbb{G}_{2m}$. Note that if we pair two elements in the group $\mathbb{G}_{m}$, we get an element in $\mathbb{G}_{2m}$ by the definition of asymmetric multilinear maps. Let $Y_i = g_{i}^{\alpha_{i}}$, where $\alpha \in \mathbb{Z}_q$. Recall from Section 2.3.2 that $x_j$ is the $j$th standard basis vector (with 1 at position $j$ and 0 at each of the other positions) and $\mathbb{G}_{x_j}$ is the $j$th source group with generator $g_{x_j}$. Also, let $X_j = g_{x_j}^{(2^j)}$ for $0 \leq j \leq m-1$, $X_m = g_{x_m}^{(2^m+1)}$ and $X'_m = g_{x_m}^{(2^m)}$. We recall the following claims from [4].

**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

$$Y_i = g_{i}^{\alpha_{i}} = g_{m}^{(\sum_{j=0}^{m-1} i_j 2^j)} = e\left( g_{x_0}^{\alpha_{0}} g_{x_0}^{1-i_0}, \cdots, g_{x_{m-1}}^{\alpha_{m-1} 1_{m-1}} g_{x_{m-1}}^{1_{m-1}} g_{x_m} \right) = e\left( X_0^{i_0} g_{x_0}^{1-i_0}, \cdots, X_{m-1}^{i_{m-1}} g_{x_{m-1}}^{1-i_{m-1}} g_{x_m} \right) \quad \-boxed{}$$

**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

$$Y_i = g_{i}^{\alpha_{i}} = g_{m}^{(\sum_{j=0}^{m-1} i'_j 2^j) + (2^m + 1)} = e\left( g_{x_0}^{\alpha_{0}'} g_{x_0}^{1-i'_0}, \cdots, g_{x_{m-1}}^{\alpha_{m-1}'} g_{x_{m-1}}^{1-i'_{m-1}} g_{x_m}^{2^m + 1} \right) = e\left( X_0^{i'_0} g_{x_0}^{1-i'_0}, \cdots, X_{m-1}^{i'_{m-1}} g_{x_{m-1}}^{1-i'_{m-1}} g_{x_m}^{2^m + 1} \right) \quad \boxed{}$$

We now make the following important observation:

**Observation 1.** Given the set of parameters $(X_0, \cdots, X_m)$ and without the knowledge of $X'_m = g_{x_m}^{(2^m)}$, it is difficult to compute the value of $Y_{N+1} = g_{x_m}^{(2^m)}$.

This follows trivially from the fact that $2^m$ cannot be expressed as the sum of other lower powers of 2.

Thus we successfully embed the parameter set $(Y_1, \cdots, Y_m, Y_{N+1}, Y_{2N})$ comprising of $O(N)$ group elements into another parameter set $(X_0, \cdots, X_m)$ comprising of $O(\log N)$ group elements, without compromising the value of the secret $Y_{N+1}$. Protecting the value of $Y_{N+1}$ from possible leakage is paramount to the security of our identity-based KAC construction, will become evident from an actual description of the construction presented next.
3.2. Our First Identity-Based KAC Construction. We present our first identity-based KAC construction that supports \( N = 2^m - 1 \) identities, where \( m \) is polynomially large in the security parameter \( \lambda \). Assume that \( \text{SetUp}''(1^\lambda, m) \) is the setup algorithm for an asymmetric multilinear map, where groups have prime order \( q \) (where \( q \) is a \( \lambda \) bit prime) and \( G_m \) is the target group. The construction is as follows.

\[
\text{SetUp}(1^\lambda, m): \text{Take as input the length } m \text{ of identities and the group order parameter } \lambda. \text{ Set } ID = \{0, 1\}^m \setminus \{0\}^m \text{ as the identity space. Let } m = \text{ the } m + 1 \text{ length vector consisting of all ones. Run } \text{SetUp}'' \text{ on } m \text{ and } 2m \text{ to obtain the public parameters } param' \text{ and } param'' \text{ for multilinear maps with target groups } G_m \text{ and } G_{2m} \text{ respectively. Choose a random } \alpha \in \mathbb{Z}_q. \text{ Set } X_j = g_{X_j}^{(2^j)} \text{ for } 0 \leq j \leq m - 1 \text{ and } X_m = g_{X_m}^{(2^{m+1})}. \text{ Next, randomly pick } \gamma \in \mathbb{Z}_q, \text{ and output the public parameter } param \text{ and the master secret key } msk \text{ as}
\]
\[
\begin{align*}
\text{param} &= (param', param'', \{X_j\}_{j \in [0, m]}, g_m^\gamma) \\
\text{msk} &= \gamma
\end{align*}
\]
Discard \( \alpha \).

\[\text{Encrypt}(param, i \in ID, M \in G_{2m}): \text{Randomly choose } t \in \mathbb{Z}_q. \text{ Recall that } Y_i = g_m^\gamma \text{ and can be computed as per the formulation in Claim 3.1 for } 1 \leq i \leq N \text{ using the multilinear map with target group } G_m. \text{ Output the ciphertext } C \text{ as}
\]
\[
C = (g_m^t, (g_m^\gamma \cdot Y_i)^t, M \cdot g_m^{\alpha(2^m)})
\]
where \( g_m^{\alpha(2^m)} \) is computed as \( e(Y_N, Y_1)^t \) using the multilinear map with target group \( G_{2m} \).

\[\text{Extract}(param, msk = \gamma, S): \text{The aggregate key for the set of identities } S \text{ is computed as:}
\]
\[
K_S = \prod_{v \in S} Y_{2m-\nu}^\gamma
\]
Again, note that \( Y_{2m-\nu} \) can be computed as per the formulation in Claim 3.1 for each \( \nu \in S \).

\[\text{Decrypt}(param, C, K_S): \text{If } i \notin S, \text{ output } \bot. \text{ Otherwise, set}
\]
\[
a_S = \left( \prod_{v \in S, v \neq i} Y_{2m-\nu+1} \right) \text{ and } b_S = \left( \prod_{v \in S} Y_{2m-\nu} \right)
\]
Parse \( C \) as \((c_0, c_1, c_2)\). Output the decrypted message as:
\[
\hat{M} = c_2 \frac{e(K_S \cdot a_S, c_0)}{e(b_S, c_1)}
\]
where the computation uses the multilinear map with target group \( G_{2m} \).

In the above KAC construction, we have assumed by default that all plaintext messages \( \hat{M} \) may be efficiently embedded as elements in the target group of the multilinear map. However, a workaround may be readily proposed. We first note that in any ciphertext output by \text{Encrypt}, the message \( \hat{M} \) is essentially multiplied with a random secret group element \( \rho \). Rather than embedding \( \hat{M} \) as a group element, we propose hashing \( \rho \) using a collision resistant hash function \( H \), and then outputting \( \hat{M} \circ H(\rho) \) in the ciphertext (here \( \circ \) denotes an appropriate operator).
3.3. Correctness. To see that the scheme is correct, that is, $\hat{M} = M$, put $c_0 = g_{t^m}$, $c_1 = (g_{t^m} Y_i)^t$ and $c_2 = M g_{t^m}^{2^{m}}$. Then we have
\[
\hat{M} = c_2 e^{(K_S \cdot a_S, c_0)} e^{(b_S, c_1)} e^{(\prod_{v \in S} Y_{2^m - v}, (g_{t^m}^v Y_i)')}
\]
\[
= c_2 e^{(\prod_{v \in S, v \neq i} Y_{2^m - v}, g_{t^m}^i)} e^{(\prod_{v \in S} Y_{2^m - v}, Y_i')}
\]
\[
= M \cdot g_{t^m}^{2^{m}} e^{(Y_{2^m}, g_{t^m}^i)}
\]
\[
= M
\]

3.4. Security (standard model). The above KAC construction is selectively CPA-secure in the standard model under the decisional $m$-HDHE assumption. More specifically, we state and prove the following theorem.

**Theorem 3.1.** Let $\text{SetUp}''$ be the setup algorithm for an asymmetric multilinear map, and let the decisional $m$-Hybrid Diffie-Hellman Exponent assumption holds for $\text{SetUp}''$. Then our proposed identity-based KAC scheme is non-adaptively CPA secure.

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

- $\text{param}'' \leftarrow \text{SetUp}''(\lambda', 2m)$ ($\lambda'$ is the adequate group size parameter for achieving $\lambda$ bit security)
- $X_j = g_{X_j}^{(2j)}$ for $0 \leq j \leq m - 1$
- $X_m = g_{X_m}^{(2m + 1)}$
- $V = g_{t^m}$ for a random $t \in \mathbb{Z}_q$ ($q$ being a $\lambda'$ bit prime)
- $Z$ is either $g_{2m}^{\alpha}$ or a random element of $\mathbb{G}_{2m}$

$B$ then proceeds as follows.

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

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

- $\text{param}$ is set as $(\text{param}''$, $\{X_j\}_{j \in \{0, \ldots, m\}}$).
- $PK$ is set as $g_{m}^{u} / Y_i$ 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 $\text{msk} = (u - \alpha')$. 

---

**Advances in Mathematics of Communications** Volume 13, No. 4 (2019), 759-778
– \( B \) then computes

\[
K_\mathcal{S} = \prod_{v \in \mathcal{S}} Y_{2^m-v}^{2^m-u-v+i}
\]

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

Since the \( g_m, \alpha, u \) and \( t \) values are chosen uniformly at random, all the parameters and the keys have an identical distribution to that in the actual construction.

**Challenge:** \( A \) picks at random two messages \( M_0 \) and \( M_1 \) from the set of possible plaintext messages in \( \mathbb{G}_{2^m} \), and provides them to \( B \). \( B \) randomly picks \( b \in \{0,1\} \), and sets the challenge as \( (C, M_0, M_1) \), where

\[
C = (V, V^u, M_b, Z)
\]

We claim that when \( Z = g_{2^m}^{t} \) (i.e. the input to \( B \) is a valid \( m \)-HDHE tuple), then \( (C, M_0, M_1) \) is a valid challenge to \( A \) as in a real attack. To see this, observe that

\[
V = g_m^t \quad \text{and} \quad V^u = (g_m^u)^t = (PK.Y_i)^t
\]

\[
M_b.Z = M_b.g_m^{t(2^m)}
\]

Thus, by definition, \( C \) is a valid encryption of the message \( M_b \) in class \( i \) and hence, \( (C, M_0, M_1) \) is a valid challenge to \( A. \)

**Guess:** The adversary \( A \) outputs a guess \( b' \) of \( b \). If \( b' = b \), \( B \) outputs 0 (indicating that \( Z = g_{2^m}^{t(2^m)} \)). Otherwise, it outputs 1 (indicating that \( Z \) is a random element in \( \mathbb{G}_{2^m} \)).

We conclude that \( B \) has the same advantage \( \epsilon \) as \( A \), which must therefore be negligible, as desired. This completes the proof of Theorem 3.4.

\[\square\]

3.5. **Security (generic group model).** We note, however, that in the absence of any secure multilinear map realizations, the HDHE assumption only holds in the generic group model. In this subsection, we present a direct proof of security for our KAC construction in the generic multilinear map model. In particular, we state the following theorem.

**Theorem 3.2.** Any generic adversary \( A \) that can make at most a polynomial number of queries to \( \text{(Encode, Mult, Pair)} \) has negligible advantage in breaking the adaptive security against chosen plaintext attacks of the KAC construction presented in Section 3.2, provided that \( 1/q^{1/2} \) is negligible in the security parameter \( \lambda. \)

**Proof.** Let \( \xi : \mathbb{Z}_q \times \mathbb{Z}^l \to \{0,1\}^n \) be the random injective function for modeling the different groups in the generic multilinear map model, as described in Section 2.6.

In particular, recall that \( \xi \) maps pairs of elements from the additive group \( \mathbb{Z}_q \) and the vector space \( \mathbb{Z}^l \) to random strings of sufficiently large length. Also, let \( A \) be an adaptive adversary under the generic model and let \( B \) be a challenger that plays the following game with \( A. \)
SetUp. Challenger $B$ sets up the system for the identity-space $ID$ consisting of all $m$ bit identities supported by the KAC construction described above. $B$ generates random $\alpha, \gamma \in \mathbb{Z}_q$. $A$ receives the following:

- $X_j = \xi(\alpha^{2^j}, 2^j)$ for $j = 0, \cdots, m - 1$
- $X_m = \xi(\alpha^{2^m+1}, 2^m)$
- $X_{m+1} = \xi(\gamma, 2^{m+1} - 1)$

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

Aggregate Key Query Phase 1. Algorithm $A$ adaptively issues aggregate key queries for any subset $\tilde{S} \subseteq ID$. $B$ responds with the aggregate key $K_{\tilde{S}}$, computed as:

$$K_{\tilde{S}} = \xi \left( \gamma \cdot \sum_{v \in \tilde{S}} \alpha^{2^{m+v} - v}, 2^{m+1} - 1 \right)$$

Commit. $A$ adaptively commits to an identity $i \in ID$, subject to the constraint that $i \notin \tilde{S}$ for all subsets $\tilde{S}$ queried in the aggregate key query phase 1.

Challenge. To make a challenge query, $A$ chooses $\hat{m}_0, \hat{m}_1 \in \mathbb{Z}_q$ and makes $\text{Encode}$ queries for the same to receive $\xi(\hat{m}_0, 2^{m+2} - 2)$ and $\xi(\hat{m}_1, 2^{m+2} - 2)$, respectively. It then provides these as the challenge messages to $B$. $B$ in turn randomly chooses $t \in \mathbb{Z}_q$ and a random $\hat{b} \in \{0, 1\}$, and sets

$$c_0 = \xi(t, 2^{m+1} - 1)$$
$$c_1 = \xi(\gamma + t \cdot \alpha^i, 2^{m+1} - 1)$$
$$c_2 = \xi(\hat{m}_b + t \cdot \alpha^{2^{m}}, 2^{m+2} - 2)$$

Finally, $B$ provides the challenge to $A$ as

$$C = (c_0, c_1, c_2)$$

and provides the challenge to $A$ as

$$(C, \xi(\hat{m}_0, 2^{m+2} - 2), \xi(\hat{m}_1, 2^{m+2} - 2))$$

where $\xi(\hat{m}_0, 2^{m+2} - 2)$ and $\xi(\hat{m}_1, 2^{m+2} - 2)$ are essentially the group element representations of the challenge message pair chosen by $A$.

Aggregate Key Query Phase 2. Algorithm $A$ continues to adaptively issue aggregate key queries for any subset $\tilde{S} \subseteq ID$ such that $i \notin \tilde{S} = \phi$. $B$ responds with the aggregate key $K_{\tilde{S}}$ as in the aggregate key query phase 1.

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

We now assume that instead of choosing random values for the set of parameters

$$(\alpha, \gamma, t, \hat{m}_0, \hat{m}_1)$$

the algorithm $B$ treats them as formal variables and maintains a list of tuples $L = \{(p, l, \epsilon)\}$, where $p$ is a polynomial in these formal variables, $l$ is the group index and $\epsilon \in \{0, 1\}^n$ for $n$ sufficiently large. The game now proceeds through the following stages:

- $B$ initializes the list with the following tuples:
\(-\left(\alpha^{2j}, 2^j, \xi_j\right)\) for randomly generated \(\xi_j \in \{0,1\}^n\), where \(j \in [0, m - 1]\)
\(-\left(\alpha^{2m+1}, \xi_m\right)\) for randomly generated \(\xi_m \in \{0,1\}^n\)
\(-\left(\alpha^{2m}, 2^{m+1} - 2, \xi_{m+1}\right)\) for randomly generated \(\xi_{m+1} \in \{0,1\}^n\)
\(-\left(\gamma, 2^{m+1} - 1, \xi_{m+2}\right)\) for a randomly generated \(\xi_{m+2} \in \{0,1\}^n\)

Thus initially, \(|L| = m + 2\). \(B\) supplies the set of strings \(\{\xi_j\}_{j \in [0, m+2]}\) to \(A\).

\(-A\) is allowed at most a polynomial number of \((\text{Encode}, \text{Mult}, \text{Pair})\) queries to which \(B\) responds as follows:

\(-\text{Encode}(x, l)\): If \(x \notin \mathbb{Z}_q\) or \(i \notin \{1, \cdots, 2^{m+2} - 2\}\), \(B\) returns \(\perp\). Otherwise, \(B\) looks for a tuple \((p_x, l, \xi) \in L\), where \(p_x\) is a constant polynomial equal to \(x\). If such a tuple exists, \(B\) returns \(\xi\). Otherwise, \(B\) generates a random \(\xi \in \{0,1\}^n\). The tuple \((p_x, l, \xi)\) is added to \(L\) and \(B\) responds with \(\xi\).

\(-\text{Mult}(\xi_k, \xi_1, b)\): \(B\) searches in \(L\) for the pair of tuples \((p_{k_0}, l_{k_0}, \xi_{k_0})\) and \((p_{k_1}, l_{k_1}, \xi_{k_1})\). If both of them exist, \(B\) returns \(\perp\). Also, if both tuples are available but \(l_{k_0} \neq l_{k_1}\), \(B\) returns \(\perp\). Otherwise, \(B\) searches in \(L\) for a tuple of the form \((p_k, l_k, \xi_k)\), where \(p_k = p_{k_0} + (-1)^b p_{k_1}\) and \(l_k \equiv l_{k_0} \equiv l_{k_1}\). If such a tuple is found, \(B\) responds with \(\xi_k\). If not, \(B\) generates a random \(\xi_k \in \{0,1\}^n\), augments \(L\) by adding the tuple \((p_k, l_k, \xi_k)\) and returns \(\xi_k\).

\(-\text{Pair}(\xi_k, \xi)\): \(B\) searches in \(L\) for the pair of tuples \((p_{k_0}, l_{k_0}, \xi_{k_0})\) and \((p_{k_1}, l_{k_1}, \xi_{k_1})\). If both of them exist, \(B\) returns \(\perp\). Also, if both tuples are available but \(l_{k_0} + l_{k_1} > 2^{m+2} - 2\), \(B\) returns \(\perp\). Otherwise, \(B\) searches in \(L\) for a tuple of the form \((p_{k}, l_{k}, \xi_{k})\), where \(p_{k} = p_{k_0} p_{k_1}\) and \(l_{k} \equiv l_{k_0} \equiv l_{k_1}\). If such a tuple is found, \(B\) responds with \(\xi_{k}\). If not, \(B\) generates a random \(\xi_{k} \in \{0,1\}^n\), augments \(L\) by adding the tuple \((p_{k}, l_{k}, \xi_{k})\) and returns \(\xi_{k}\).

We stress the fact 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 \(j \geq 2^{m+2} - 2\). Note that \(B\) can make the string responses \(\xi\) to \(A\) arbitrarily long, and thus, hard to guess. Hence, without loss of generality, we assume that all \text{Mult} and \text{Pair} queries made by \(A\) are precisely on strings furnished by \(B\).

- Algorithm \(A\) is allowed to adaptively issue aggregate key queries for any subset \(\hat{S} \subseteq TD\). In response, \(B\) adds the tuple

\[
\left(\gamma \cdot \sum \alpha^{2m-v}, 2^{m+1} - 1, \xi\right)
\]

for a randomly generated \(\xi \in \{0,1\}^n\). The same \(\xi\) is given as response to \(A\).

- \(A\) adaptively commits to an identity \(i \in TD\), subject to the constraint that \(i \notin \hat{S}\) for all subsets \(\hat{S}\) queried in the aggregate key query phase 1.

- \(A\) makes a single encryption query on the identity \(i \in S\). \(B\) creates a new formal variables \(t, \hat{m}, \hat{m}_0, \hat{m}_1\) and adds the following tuples to its list \(L\):

\(-\left(t, 2^{m+1} - 1, \xi_0\right)\)
\(-\left(\gamma + t \cdot \alpha^i, 2^{m+1} - 1, \xi_1\right)\)
\(-\left(\hat{m} + t \cdot \alpha^m, 2^{m+2} - 2, \xi_2\right)\)
\(-\left(\hat{m}_0, 2^{m+2} - 2, \xi_3\right)\)
\(-\left(\hat{m}_1, 2^{m+2} - 2, \xi_4\right)\)

Once again \(\xi_j^*\) for \(j \in \{0, \cdots, 4\}\) are randomly generated strings in \(\{0,1\}^n\) that are provided to \(A\) as the challenge vector.
A may continue to issue adaptive oracle queries and aggregate key queries. B responds as above.

- A outputs a random \( \hat{b}' \in \{0, 1\} \).
- At this point, B chooses random values for the variables \( \alpha, \gamma, t \) and asks A for two random message inputs \( m_0 \) and \( m_1 \), to set \( \hat{m}_0 = m_0 \), and \( \hat{m}_1 = m_1 \). B also randomly chooses \( \hat{b} \in \{0, 1\} \) and sets \( \hat{m} = m_{\hat{b}} \).

We now look at the probability that A wins the game. We denote by \( \Upsilon \) a false polynomial equality event where, for two random tuples \((p, l, \xi)\) and \((p', l', \xi')\) in the list \( L \) such that \( l = l' \), we have \( p(\alpha_{j,b}, \cdots) = p'(\alpha_{j,b}, \cdots) \) even though \( p \neq p' \). In case an instance of \( \Upsilon \) occurs, B fails to simulate the oracle perfectly. We say that A wins the game if either \( \hat{b} = \hat{b}' \) or an instance of the event \( \Upsilon \) occurs.

We now look at the probability that a random choice of values for the formal variables

\[
(\alpha, \gamma, t, m_0, m_1) \in \mathbb{Z}_q^{(5)}
\]

and a random choice of \( \hat{b} \in \{0, 1\} \) results in the event \( \Upsilon \). First, we make the following straightforward observations.

**Observation 2.** The maximum degree of any polynomial in \( L \) is at most \( 2^{m+1} \).

**Observation 3.** Substituting the formal variable \( \hat{m}_0 \) with \( m_0 \), \( \hat{m}_1 \) with \( m_1 \), and \( \hat{m} \) with \( m_{\hat{b}} \) results in an instance of the false polynomial equality event \( \Upsilon \) with negligible probability.

It follows from the Schwartz-Zippel lemma [15] that the probability that a randomly chosen pair of polynomials in \( L \) evaluate to the same value for a random choice of variable values, is upper bounded by \((2^{m+1})/q \). Next, assume that A makes \( Q_E \) queries to \texttt{Encode}, \( Q_M \) queries to \texttt{Mult} and \( Q_P \) queries to \texttt{Pair} during Oracle Query Phase, and a total of \( Q_K \) queries in Aggregate Key Query Phase 1 and Aggregate Key Query Phase 2. Table 1 summarizes the maximum possible contributions to \( |L| \) by the tuples added by B at different query stages. Note that \( |L| \) is upper bounded by \((Q_E + Q_M + Q_P + 3m + 8) \). It easily follows that the probability of a false polynomial equality event \( \Upsilon \) is upper bounded

\[
Pr(\Upsilon) \leq (Q_E + Q_M + Q_P + Q_K + m + 7)^2 (2^{m+1})/2q
\]

If the event \( \Upsilon \) does not occur, B simulates the oracle in response to A’s queries perfectly and, from A’s view, \( \hat{b} \) is independent as it was chosen after the simulation. Hence we have

\[
Pr[\hat{b} = \hat{b}' \mid \Upsilon] = 1/2
\]

This in turn gives us the following relations:

\[
Pr[\hat{b} = \hat{b}' \mid \Upsilon]Pr[\Upsilon] = \frac{1 - Pr[\Upsilon]}{2}
\]

\[
Pr[\hat{b} = \hat{b}' \mid \Upsilon]Pr[\Upsilon] + Pr[\Upsilon] = \frac{1 + Pr[\Upsilon]}{2}
\]

From this, it is straightforward to conclude that the advantage of the generic adversary A may be upper bounded as \((Q_E + Q_M + Q_P + Q_K + m + 7)^2 (2^{m+1})/4q \). For \( Q_E, Q_M, Q_P, Q_K, m \) polynomial in the security parameter \( \lambda \), this quantity is negligible provided that \( 1/q^{1/2} \) is negligible, or in particular, \( q \approx 2^{2\lambda} \). This completes the proof of Theorem 3.5.
Table 1. Theorem 1: Upper Bounds on Contributions to Length of $L$

| Query Stage                                   | Maximum Contribution to $|L|$ |
|-----------------------------------------------|-------------------------------|
| Setup                                         | $m + 2$                       |
| Oracle Query Phase                            | $Q_E + Q_M + Q_P$             |
| Aggregate Key Query Phase (1 and 2)           | $Q_K$                         |
| Challenge                                     | 5                             |
| **Total**                                     | $Q_E + Q_M + Q_P + Q_K + m + 7$ |

4. **KAC using symmetric multilinear maps with tighter security bounds**

We have demonstrated that the KAC construction described in Section 3 is secure in the generic multilinear map model. However, the presence of high degree exponents such as $\alpha^{2^m}$ in this construction means that the adversary can construct polynomials with degree as high as $\approx 2^m$ in the secret $\alpha$. Hence, the aforementioned constructions require the group order parameter $\lambda' \approx 2\lambda$ (instead of the normal $\lambda$) or $q = 2^{3\lambda}$ to achieve $\lambda$-bit security.

This motivates us to present a second KAC construction that is adaptively secure in the generic multilinear map model for $q = 2^\lambda$. This construction is inspired by techniques from [4]. It uses symmetric multilinear maps, as opposed to asymmetric multilinear maps used in the first construction.

4.1. **The basic idea.** Our second construction uses a set of public parameters that are not derived from a single scalar $\alpha$. Instead, each parameter is derived from a separate random scalar, to avoid the high degree exponents in the previous constructions. The trick of embedding linearly many group elements in a logarithmic size public parameter is still used. However, the presence of multiple scalars implies that we cannot use the same embedding technique used in the first construction.

Instead, we adopt a strategy presented in [4] of using a Naor-Reingold-style PRF [16]. The ciphertext in this construction cannot consist of $O(1)$ group elements, since that would make decryption impossible. At the same time, we cannot allow the ciphertext size to exceed polylogarithmic, since that would render the KAC construction inefficient.

To achieve this, we ensure that the necessary ciphertext components for decryption are the outputs of another Naor-Reingold-style PRF, and the decryptor can generate precisely these outputs by puncturing the PRF at suitable points, similar to as described in [3] and [4].

4.2. **The construction.** Let $Setup'(1^\lambda, m)$ be the setup algorithm for an $m$-linear map with groups of prime order $q$ ($q$ being a $\lambda$ bit prime) and the target group $G_m$. Our second identity-based KAC consists of the following algorithms.

SetUp($1^\lambda, m$): Set up the KAC system for $TD$ consisting of all $m$ bit identities. Let $param' \leftarrow Setup'(1^\lambda, m + 1)$ be the public parameters for a symmetric multilinear map, with $G_{m+1}$ being the target group. For $j = 0, \cdots, m-1$ and $b = 0, 1$, generate random $\alpha_{j,b} \in Z_q$ and let $X_{j,b} = g_{1,\cdot}^{\alpha_{j,b}}$. Finally, pick $\gamma \in Z_q$, and output the public parameter $param$ and the master secret key $msk$ as:

\[
\begin{align*}
param &= (param', \{X_{j,b}\}_{j \in \{0, \cdots, m-1\}, b \in \{0,1\}, g_m^\gamma}) \\
msk &= \gamma
\end{align*}
\]
Discard all $\alpha_{j,b}$ after $\text{param}$ has been output.

**Claim 4.1.** For any identity $i \in ID$, one can compute $Y_i = g_m^{\prod_{j=0}^{m-1} \alpha_{j,i_j}}$ where $i_j$ is the $j$-th 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.

**Encrypt**($\text{param}, i, M$): Randomly choose $t, x \in \mathbb{Z}_q$ and set

$$
c_0 = g_1^t
$$
$$
c_{j+1} = X_{j, (1-i_j)}^t \text{ for } j = 0, 1, \cdots, m - 1
$$
$$
c_{m+1} = g_m^x \cdot Y_i^t
$$
$$
c_{m+2} = M \cdot (g_m^{\gamma x})^t = M \cdot (e(g_m^\gamma, g_1^x))^t
$$

Note that $i_j$ denotes the $j$-th bit in the binary representation of $i$. Finally, output the ciphertext as

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

**Extract**($\text{param}, \text{msk} = \gamma, S$): The aggregate key for the subset $S$ is computed as:

$$
K_S = \left(\prod_{v \in S} Y_v\right)^\gamma \cdot g_1^\gamma
$$

**Decrypt**($\text{param}, C, K_S$): If $i \notin S$, output $\perp$. Otherwise, use the result from Claim 4.2 to compute $Y_v^t$ for each $v \in S \setminus \{i\}$, and set

$$
a_S = \left(\prod_{v \in S, v \neq i} Y_v\right)^t
$$

We argue that $a_S$ can be computed by making the following claim.

**Claim 4.2.** For any identity $v \neq i$, one can compute $Y_v^t$ given $C$.

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

$$
Y_v^t = e(\{X_{j,v_j}\}_{j \in \{0, \cdots, m-1\} \setminus \{j'\}}, c_{j'+1}^t)
$$

because:

$$
Y_v^t = g_m^{t \prod_{j=0}^{m-1} \alpha_{j,v_j}}
$$
$$
= g_m^{t \alpha_{j',v_{j'}} \prod_{j \neq j'} \alpha_{j,v_j}}
$$
$$
= g_m^{t \alpha_{j',1-i_{j'}} \prod_{j \neq j'} \alpha_{j,v_j}}
$$
$$
= e(\{X_{j,v_j}\}_{j \in \{0, \cdots, m-1\} \setminus \{j'\}}, c_{j'+1}^t)
$$

Note that for a given $i$, each $Y_v^t$ for $v \neq i$ and random $t$ is also the output of a Naor-Reingold style PRF. Moreover, the PRF is punctured at $i$ to generate $Y_v^t$ for $v \neq i$ without the knowledge of $Y_i^t$. 
Let $C = (c_0, \cdots, c_{m+1}, c_{m+2})$ and $K_{S} = (K_1, K_2)$. Output the decrypted message as

$$\hat{M} = c_{m+2} \frac{e(K_1, c_0)}{e(c_{m+1}, a_{S}, K_2)}$$

Once again, we have assumed by default that the ciphertext $C$ embeds the information about the identity $i$ it corresponds to, and $K_S$ embed the information about the subset $S$ it corresponds to. Observe that in this construction, no high-degree terms are generated hence the system is secure for a group size parameter $q = 2^{\lambda}$ instead of $2^{2\lambda}$ in the first construction ($\lambda$ being the security parameter). Also, the total multilinearity in handling $2^\lambda$ identities is $\lambda + 1$, as compared to $2\lambda$ in the previous construction. On the flip side, the ciphertext size for this construction is $O(\log N)$ group elements, as compared to $O(1)$ group elements in the previous constructions.

4.2.1. Correctness. The correctness of this scheme may be easily shown as follows:

$$\hat{M} = c_{m+2} \frac{e(K_1, c_0)}{e(c_{m+1}, a_{S}, K_2)}$$

$$= c_{m+2} \frac{e((\prod_{v \in S} Y_v)^\gamma, g_1^i)}{e(g_{m+1}^x, \left(Y_i \prod_{v \in S, v \neq i} Y_v \right)^t, g_1^\gamma)}$$

$$= c_{m+2} \frac{e(g_{m+1}^x, g_1^\gamma)}{e(g_{m+1}^x, g_1^\gamma)}$$

$$= \hat{M}$$

4.3. Security. The above KAC construction is provably secure in the generic multilinear map model in the adaptive setting, with tighter bounds on the group order parameters as compared to the previous constructions. In particular, we state the following theorem.

**Theorem 4.1.** Any generic adversary $A$ that can make at most a polynomial number of queries to $(\text{Encode}, \text{Mult}, \text{Pair})$ has negligible advantage in breaking the adaptive security of the KAC construction presented in Section 4.2, provided that $1/q$ is negligible, for $q \approx 2^\lambda$, where $\lambda$ is the security parameter.

**Proof.** Let $\xi : \mathbb{Z}_q \times \mathbb{Z}^l \rightarrow \{0, 1\}^n$ be the random injective function for modeling the different groups in the generic multilinear map model, as described in Section 2.6. In particular, recall that $\xi$ maps pairs of elements from the additive group $\mathbb{Z}_q$ and the vector space $\mathbb{Z}^l$ to random strings of sufficiently large length. Also, let $A$ be an adaptive adversary under the generic model and let $B$ be a challenger that plays the following game with $A$:

**SetUp:** Challenger $B$ sets up the system for the identity-space $\mathcal{ID}$ consisting of all $m$ bit identities supported by the KAC construction described above. $B$ generates random $a_{j, b} \in \mathbb{Z}_q$ for $j = 0, \cdots, m - 1$, and $b = 0, 1$. $B$ also generates random $\gamma, x \in \mathbb{Z}_q$. $A$ receives the following:

- $X_{j, b} = \xi(a_{j, b}, 1)$ for $j = 0, \cdots, m - 1$, and $b = 0, 1$
- $X_m = (\xi(\gamma, m))$

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

---

**Advances in Mathematics of Communications**
Volume 13, No. 4 (2019), 759–778
Aggregate Key Query Phase 1. Algorithm \( A \) adaptively issues aggregate key queries for any subset \( S \subseteq \mathcal{I} \mathcal{D} \). \( B \) responds with the aggregate key \( K_S \), computed as:

\[
K_S = \left( \xi \left( \gamma \sum_{v \in S} \alpha_{j,vj} \right), \xi (\gamma,1) \right)
\]

Commit. \( A \) adaptively commits to an identity \( i \in \mathcal{I} \mathcal{D} \), subject to the constraint that \( i \notin S \) for all subsets \( S \) queried in the aggregate key query phase 1.

Challenge. To make a challenge query, \( A \) chooses \( \hat{m}_0, \hat{m}_1 \in \mathbb{Z}_q \) and makes \textsf{Encode} queries for the same to receive \( \xi (\hat{m}_0, m+1) \) and \( \xi (\hat{m}_1, m+1) \), respectively. It then provides these as the challenge messages to \( B \). \( B \) in turn randomly chooses \( t, x \in \mathbb{Z}_q \) and a random \( \hat{b} \in \{0,1\} \), and sets

\[
\begin{align*}
c_0 &= \xi (t,1) \\
c_{j+1} &= \xi (t\alpha_{j,(1-i_j)},m) \quad \text{for } j = 0, 1, \ldots, m-1 \\
c_{m+1} &= \xi (x+t \cdot \prod_{j=0}^{m-1} \alpha_{j,i_j}, m) \\
c_{m+2} &= \xi (\hat{m}_b + \gamma xt, m+1)
\end{align*}
\]

Finally, \( B \) provides the challenge to \( A \) as:

\( \mathcal{C} = \{c_j \}_{j=0}^{m+2} \)

Guess: \( A \) outputs a guess \( \hat{b}' \) of \( \hat{b} \). If \( \hat{b}' = \hat{b} \), \( A \) wins the game.

We now assume that instead of choosing random values for the set of parameters

\[
\{ (\alpha_{j,b})_{j=0}^{m-1}, b \in \{0,1\} \}, \gamma, t, x, \hat{m}_0, \hat{m}_1
\]

the algorithm \( B \) treats them as formal variables and maintains a list of tuples \( L = \{ (p,l,\epsilon) \} \), where \( p \) is a polynomial in these formal variables, \( l \) is the group index and \( \epsilon \in \{0,1\}^n \). The game now proceeds through the following stages:

- \( B \) initializes the list with the following tuples:
  - \( (\alpha_{j,b}, 1, \xi_{2j+b}) \) for randomly generated \( \xi_{2j+b} \in \{0,1\}^n \), where \( j \in \{0, \ldots, m-1\} \) and \( b \in \{0,1\} \)
  - \( (\gamma, m, \xi_{2m}) \) for a randomly generated \( \xi_{2m} \in \{0,1\}^n \)

Thus initially, \( |L| = 2m+1 \). \( B \) supplies the set of strings \( \{\xi_j\}_{j=0, \ldots, 2m+1} \) to \( A \).

- \( A \) is allowed at most a polynomial number of \textsf{(Encode,Mult,Pair)} queries to which \( B \) responds as follows:
  - \textsf{Encode}(x,l): If \( x \notin \mathbb{Z}_q \) or \( i \notin \{1, \ldots, m+1\} \), \( B \) returns \( \bot \). Otherwise, \( B \) looks for a tuple \( (p_x,l,\xi) \in L \), where \( p_x \) is a constant polynomial equal to \( x \). If such a tuple exists, \( B \) returns \( \xi \). Otherwise, \( B \) generates a random \( \xi \in \{0,1\}^n \). The tuple \( (p_x,l,\xi) \) is added to \( L \) and \( B \) responds with \( \xi \).
  - \textsf{Mult}(\xi_k,\xi_{k+1},b): \( B \) searches in \( L \) for the pair of tuples \( (p_{k_0},l_{k_0},\xi_{k_0}) \) and \( (p_{k},l_{k},\xi_{k}) \). Unless both of them exist, \( B \) returns \( \bot \). Also, if both tuples are available but \( l_{k_0} \neq l_k, \) \( B \) returns \( \bot \). Otherwise, \( B \) searches in \( L \) for a tuple of the form \( (p_k,l_k,\xi_k) \), where \( p_k = p_{k_0} + (-1)^k p_{k_1} \) and \( l_k \equiv l_{k_0} \equiv l_{k_1} \). If such a tuple is found, \( B \) responds with \( \xi_k \). If not, \( B \) generates a random \( \xi_k \in \{0,1\}^n \), augments \( L \) by adding the tuple \( (p_k,l_k,\xi_k) \) and returns \( \xi_k \).
A

- **Pair**($\xi_0, \xi_1$): $B$ searches in $L$ for the pair of tuples $(p_{k_0}, l_{k_0}, \xi_{k_0})$ and $(p_{k_1}, l_{k_1}, \xi_{k_1})$. Unless both of them exist, $B$ returns $\perp$. Also, if both tuples are available but $l_{k_0} + l_{k_1} > m + 1$, $B$ returns $\perp$. Otherwise, $B$ searches in $L$ for a tuple of the form $(p'_k, l'_k, \xi'_k)$, where $p'_k = p_{k_0} \cdot p_{k_1}$ and $l'_k = l_{k_0} + l_{k_1}$.

If such a tuple is found, $B$ responds with $\xi'_k$. If not, $B$ generates a random $\xi'_k \in \{0, 1\}^n$, augments $L$ by adding the tuple $(p'_k, l'_k, \xi'_k)$ and returns $\xi'_k$.

We stress the fact 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 $j > m + 1$. Note that $B$ can make the string responses $\xi$ to $A$ arbitrarily long, and thus, hard to guess. Hence, without loss of generality, we assume that all **Mult** and **Pair** queries made by $A$ are precisely on strings furnished by $B$.

- Algorithm $A$ is allowed to adaptively issue broadcast aggregate key queries for any subset $S \subseteq TD_1$. In response, $B$ adds the tuples

$$\left(\frac{\gamma}{m} \sum_{v \in S} \prod_{j=0}^{m-1} a_{j, v_j}, m, \xi'_1\right)$$

and

$$(\gamma, 1, \xi'_2)$$

for randomly generated $\xi'_1, \xi'_2 \in \{0, 1\}^n$. The same $\xi'_1$ and $\xi'_2$ are given as response to $A$.

- $A$ commits to a set $S$ such that $S \cap \tilde{S} = \phi$ for all subsets $\tilde{S}$ queried in the first aggregate key query phase. $B$ chooses $i \in S$ and communicates the same to $A$.

- $A$ makes a single encryption query on the identity $i \in S$. $B$ creates a new formal variables $t, \tilde{m}, \tilde{m}_0, \tilde{m}_1$ and adds the following tuples to its list $L$:

- $(t, 1, \xi^*)$
- $(t \alpha_{j, (1-i_j)}, 1, \xi'_{j})$ for $j = 0, 1, \cdots, m - 1$
- $(\tilde{m} + t \cdot \prod_{j=0}^{m-1} \alpha_{j, i_j}, m, \xi^*)$
- $(\tilde{m} + \gamma x t, m + 1, \xi^*_{m+1})$
- $(\tilde{m}0, m + 1, \xi^*_{m+2})$
- $(\tilde{m}1, m + 1, \xi^*_{m+3})$

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

- $A$ may continue to issue adaptive oracle queries and aggregate key queries. $B$ responds as above.

- $A$ outputs a random $\hat{b} \in \{0, 1\}$.

- At this point, $B$ chooses random values for $\alpha_{j, b}, \gamma, x, t$ and asks $A$ for two random input messages $m_0$ and $m_1$ to set $\tilde{m}_0 = m_0$, and $\tilde{m}_1 = m_1$. $B$ also randomly chooses $\hat{b} \in \{0, 1\}$ and sets $\tilde{m} = \tilde{m}_b$.

Recall that we denoted by $\Upsilon$ a *false polynomial equality* event where, for two random tuples $(p, l, \xi)$ and $(p', l', \xi')$ in the list $L$ such that $l = l'$, we have $p(\alpha_{j, b}, \cdots) = p'(\alpha_{j, b}, \cdots)$ even though $p \neq p'$. In case an instance of $\Upsilon$ occurs, $B$ fails to simulate the oracle perfectly. We say that $A$ wins the game if $\hat{b} = \hat{b}'$ or an instance of the event $\Upsilon$ occurs.

We now look at the probability that a random choice of values for the formal variables

$$(\alpha_{j, b}, \gamma, x, t, m_0, m_1) \in \mathbb{Z}_q^{2m+5}$$

**Advances in Mathematics of Communications**

**Volume 13, No. 4 (2019), 759–778**
and a random choice of $\hat{b} \in \{0, 1\}$ results in the event $\Upsilon$. We make the following straightforward observations.

**Observation 4.** The maximum degree of any polynomial in $L$ is at most $m + 1$.

**Observation 5.** Substituting the formal variable $\hat{m}_0$ with $m_0$, $\hat{m}_1$ with $m_1$, and $\hat{m}$ with $m_\hat{b}$ results in an instance of the false polynomial equality event $\Upsilon$ with negligible probability.

It follows from the Swartz-Zippel lemma [15] that the probability that a randomly chosen pair of polynomials in $L$ evaluate to the same value for a random choice of variable values, is upper bounded by $(m + 1)/q$. Next, 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**, and a total of $Q_K$ queries in **Aggregate Key Query Phase 1** and **Aggregate Key Query Phase 2**. Table 2 summarizes the maximum possible contributions to $|L|$ by the tuples added by $B$ at different query stages. Note that $|L|$ is upper bounded by $(Q_E + Q_M + Q_P + 2Q_K + 3m + 6)$. It easily follows that the probability of a false polynomial equality event $\Upsilon$ is upper bounded as

$$Pr(\Upsilon) \leq (Q_E + Q_M + Q_P + 3m + 8)^2 (m + 1)/2q$$

If the event $\Upsilon$ does not occur, $B$ simulates the oracle in response to $\mathcal{A}$’s queries perfectly and, from $\mathcal{A}$’s view, $\hat{b}$ is independent as it was chosen after the simulation. Hence, using a similar analysis as in the proof of Theorem 1, it is straightforward to conclude that the advantage of the generic adversary $\mathcal{A}$ may be upper bounded as $(Q_E + Q_M + Q_P + 2Q_K + 3m + 6)^2 (m + 1)/4q$. For $Q_E, Q_M, Q_P, Q_K, m$ polynomial in the security parameter $\lambda$, this quantity is negligible provided that $1/q$ is negligible, or in particular, $q \approx 2^\lambda$, as desired. This completes the proof of Theorem 4.3.

| Query Stage                        | Maximum Contribution to $|L|$          |
|-----------------------------------|----------------------------------------|
| SetUp                             | $2m + 1$                                |
| Oracle Query Phase                | $Q_E + Q_M + Q_P$                       |
| Aggregate Key Query Phase (1 and 2)| $2Q_K$                                  |
| Challenge                         | $m + 5$                                 |
| Total                             | $Q_E + Q_M + Q_P + 2Q_K + 3m + 6$      |

5. **Conclusion**

In this paper, we presented the first identity-based constructions for the key-aggregate cryptosystem (KAC) using multilinear maps. Our constructions achieve poly-logarithmic upper bounds on the overhead for the system parameters and the ciphertexts, and constant upper bounds on the size of the aggregate key. The crux of our constructions lies in the use of multilinear maps to embed polynomially many parameter elements in the target multilinear group, using only logarithmically many elements in the source multilinear groups. This in turn allows us to only publish the logarithmic-size parameter set, making the constructions identity-based.

Our first construction uses asymmetric multilinear maps, and is secure in the generic multilinear map model with full collusion resistance against any number of colluding parties. Our second construction uses symmetric multilinear maps,
and affords tighter bounds on the group size parameters for security in the generic multilinear map model, as compared to the first construction. It also has a lesser requirement in the degree of multilinearity. The trade-off is in the size of the ciphertext, which is logarithmic in the second construction, as opposed to constant in the first. To the best of our knowledge, these are the first identity-based KAC constructions in the cryptographic literature.

References

[1] D. Boneh, X. Boyen and E.-J. Goh, Hierarchical identity based encryption with constant size ciphertext, in Advances in Cryptology–EUROCRYPT 2005, Springer, 3494 (2005), 440–456.

[2] D. Boneh, C. Gentry and B. Waters, Collusion resistant broadcast encryption with short ciphertexts and private keys, in Advances in Cryptology–CRYPTO 2005, Springer, 3621 (2005), 258–275.

[3] D. Boneh and B. Waters, Constrained pseudorandom functions and their applications, in Advances in Cryptology–ASIACRYPT 2013, Springer, 8270 (2013), 280–300.

[4] D. Boneh, B. Waters and M. Zhandry, Low overhead broadcast encryption from multilinear maps, in Advances in Cryptology–CRYPTO 2014, Springer, 8616 (2014), 206–223.

[5] D. Boneh, D. J. Wu and J. Zimmerman, Immunizing multilinear maps against zeroizing attacks, IACR Cryptology ePrint Archive, 2014 (2014), 930.

[6] J. H. Cheon, K. Han, C. Lee, H. Ryu and D. Stehlé, Cryptanalysis of the multilinear map over the integers, in Advances in Cryptology–EUROCRYPT 2015, Springer, 9056 (2015), 3–12.

[7] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou and R. H. Deng, Key-aggregate cryptosystem for scalable data sharing in cloud storage, Parallel and Distributed Systems, IEEE Transactions on, 25 (2014), 468–477.

[8] J. Coron, M. S. Lee, T. Lepoint and M. Tibouchi, Cryptanalysis of GGH15 multilinear maps, in Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II, 9815 (2016), 607–628.

[9] J.-S. Coron, T. Lepoint and M. Tibouchi, Practical multilinear maps over the integers, in Advances in Cryptology–CRYPTO 2013, Springer, 8042 (2013), 476–493.

[10] J.-S. Coron, T. Lepoint and M. Tibouchi, Cryptanalysis of two candidate fixes of multilinear maps over the integers, IACR Cryptology ePrint Archive, 2014 (2014), 975.

[11] S. Garg, C. Gentry and S. Halevi, Candidate multilinear maps from ideal lattices, Advances in cryptology–EUROCRYPT 2013, 7881 (2013), 1–17.

[12] C. Gentry, S. Gorbunov and S. Halevi, Graph-induced multilinear maps from lattices, in Theory of Cryptography, Springer, 9015 (2015), 498–527.

[13] C. Gentry, S. Halevi, H. K. Maji and A. Sahai, Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero, IACR Cryptology ePrint Archive, 2014 (2014), 929.

[14] J. Kilian, Founding cryptography on oblivious transfer, in Proceedings of the twentieth annual ACM symposium on Theory of computing, ACM, 1988, 20–31.

[15] D. Moshkovitz, An alternative proof of the schwartz-zippel lemma., in Electronic Colloquium on Computational Complexity (ECCC), 17 (2010), 34.

[16] M. Naor and O. Reingold, Number-theoretic constructions of efficient pseudo-random functions, Journal of the ACM (JACM), 51 (2004), 231–262.

[17] S. Patranabis, Y. Shrivastava and D. Mukhopadhyay, ymnic key-aggregate cryptosystem on elliptic curves for online data sharing, in Progress in Cryptology–INDOCRYPT 2015, Springer, 9462 (2015), 25–44.

[18] S. Patranabis, Y. Shrivastava and D. Mukhopadhyay, Provably secure key-aggregate cryptosystems with broadcast aggregate keys for online data sharing on the cloud, IEEE Trans. Computers, 66 (2017), 891–904.

Received October 2018; revised March 2019.

E-mail address: sikhar.patranabis@iitkgp.ac.in
E-mail address: debdeep@cse.iitkgp.ac.in