Decentralized Multi-Authority Attribute-Based Inner-Product FE: Large Universe and Unbounded

Pratish Datta[1] and Tapas Pal[2]

[1] NTT Research, USA
pratish.datta@ntt-research.com
[2] NTT Social Informatics Laboratory, Tokyo, Japan
tapas.pal.wh@hco.ntt.co.jp

April 20, 2023

Abstract

This paper presents the *first* decentralized multi-authority attribute-based inner product functional encryption (MA-ABIPFE) schemes supporting vectors of a priori unbounded lengths. The notion of AB-IPFE, introduced by Abdalla et al. [ASIACRYPT 2020], combines the access control functionality of attribute-based encryption (ABE) with the possibility of evaluating linear functions on encrypted data. A decentralized MA-ABIPFE defined by Agrawal et al. [TCC 2021] essentially enhances the ABE component of AB-IPFE to the decentralized multi-authority setting where several authorities can independently issue user keys involving attributes under their control. In MA-ABIPFE for unbounded vectors (MA-ABUIPFE), encryptors can encrypt vectors of arbitrary length under access policies of their choice whereas authorities can issue secret keys to users involving attributes under their control and vectors of arbitrary lengths. Decryption works in the same way as for MA-ABIPFE provided the lengths of the vectors within the ciphertext and secret keys match.

We present two MA-ABUIPFE schemes supporting access policies realizable by linear secret sharing schemes (LSSS), in the significantly faster prime-order bilinear groups under decisional assumptions based on the target groups which are known to be weaker compared to their counterparts based in the source groups. The proposed schemes demonstrate different trade-offs between versatility and underlying assumptions. The first scheme allows each authority to control a bounded number of attributes and is proven secure under the well-studied decisional bilinear Diffie-Hellman (DBDH) assumption. On the other hand, the second scheme allows authorities to control exponentially many attributes, that is, supports large attribute universe, and is proven secure under a non-interactive q-type variant of the DBDH assumption called L-DBDH, similar to what was used in prior large-universe multi-authority ABE (MA-ABE) construction.

When compared with the only known MA-ABIPFE scheme due to Agrawal et al. [TCC 2021], our schemes offer significantly higher efficiency while offering greater flexibility and security under weaker assumptions at the same time. Moreover, unlike Agrawal et al., our schemes can support the appearance of the same attributes within an access policy arbitrarily many times. Since efficiency and practicality are the prime focus of this work, we prove the security of our constructions in the random oracle model against static adversaries similar to prior works on MA-ABE with similar motivations and assumptions. On the technical side, we extend the unbounded IPFE techniques of Dufour-Sans and Pointcheval [ACNS 2019] to the context of MA-ABUIPFE by introducing a novel *hash-decomposition* technique.

*This is the full version of an extended abstract that appears in the proceedings of PKC 2023.

Table of Contents

1 Introduction

Functional encryption (FE), introduced by Boneh, Sahai and Waters [BSW11] and O'Neill [O'N10] is an advanced form of public key encryption (PKE) designed for computing on encrypted data while maintaining its confidentiality beyond the computed results. FE delivers cryptographic solutions to a wide variety of privacy-enhancing technologies from enabling finer access control to outsourcing computations on sensitive data to the cloud. Starting with the work of Abdalla et al. [ABCP15], a long sequence of works [ABCP16, ALS16, DDM16, ABKW19, Tom19] studied FE schemes for the class of linear functions, also known as inner product FE (IPFE). In IPFE, the ciphertexts and functional secret keys are associated with vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively while a decrypter only learns the inner product $\boldsymbol{x} \cdot \boldsymbol{y}$ and nothing else about $\boldsymbol{x}$. Although the functionality is simple, IPFE has found a great amount of applications in both theory, for example, designing more expressive FE schemes for quadratic [JLS19, Gay20] and general functions [JLMS19, JLS21] and in practice, for example, performing statistical studies on encrypted data, evaluating polynomials, computing conjunctions and disjunctions [ABCP15], or calculating hamming weights in biometric authentications [LKK+18, ZR18], constructing trace and revoke schemes [ABP+17]. However, any IPFE system suffers from an inherent leakage of data due to it's linear functionality. In fact, releasing a set of secret keys for vectors forming a basis of the underlying vector space would result in a complete break of the system since it enables the recovery of the master secret key of the IPFE system and hence uncover all the encrypted data in the system.

One natural way to control such leakage of data in IPFE is to combine it with attribute-based encryption (ABE), that is, to additionally associate access policies/attributes within the ciphertexts/secret keys (or the other way around) in the same spirit as attribute-based encryption (ABE) such that the eligibility for computing on the encrypted data requires a prior validation of the attributes by the policy. Such access control mechanism in IPFE was introduced by Abdalla et al. [ACGU20] where they termed this upgraded notion as *attribute-based* IPFE (AB-IPFE). The notion of AB-IPFE [ACGU20, AGT21a, PD21] has been mostly explored in the setting where a single authority is responsible for managing all the attributes in the system and issuing secret keys to users. This not only is a limitation from the point of view of trust, but also it is problematic for practical applications. In fact, in reality, different attributes are governed by different authorities, for example, academic degrees are handled by universities, medical attributes are managed by hospitals while driving licenses are controlled by transportation or automobile agencies.

Multi Authority AB-IPFE: Inspired by the notion of *multi-authority* ABE (MA-ABE) [LW11, RW15, OT20, DKW21a, DKW21b, DKW22, WWW22] which deals with the decentralization of attribute management in the context of ABE, Agrawal et al. [AGT21b] initiated the study of *multi-authority* AB-IPFE (MA-ABIPFE) which enhances the ABE segment of AB-IPFE to the multi-authority setting. That is, just like MA-ABE, in MA-ABIPFE individual authorities are allowed to generate their own master key pairs and provide secret keys for attributes *only* under their control without interacting with the other authorities. A user learns $\boldsymbol{x} \cdot \boldsymbol{y}$ by decrypting a ciphertext generated with respect to a policy P and a vector $\boldsymbol{x}$ using various secret keys associated to a vector $\boldsymbol{y}$ and the different attributes it possesses that are obtained from the authorities controlling those attributes. Some potential practical application of MA-ABIPFE could be computing average salary of employees in an organization possessing a driving license and holding a Ph.D, statistics determining mental health of students of different departments in a university, etc.

Despite its countless potential applications, so far the only candidate MA-ABIPFE scheme, is due to Agrawal et al. [AGT21b] which supports access policies realizable by linear secret sharing schemes (LSSS) and is designed in a composite-order group and the security is based on variants of the subgroup decision assumptions which are source group assumptions, that is, assumptions made about the source groups of the underlying bilinear pairing. It is a well-known fact that composite-order bilinear groups are very expensive both in terms of computation and communication/storage.

This is reflected in the MA-ABIPFE of [AGT21b], especially the decryption takes an unacceptable time of around five days (as shown in Table 1.2) when run using reasonable parameters, which clearly makes the scheme impractical. In order to address this efficiency bottleneck, a possible way to avoid this heavy efficiency bottleneck is to look for a construction in the prime-order bilinear groups which are way better in terms of the above parameters compared to their composite-order counterparts [Fre10, Lew12, Gui13].

Another significant drawback of the MA-ABIPFE is that the vector lengths are fixed and the number of authorities or attributes are bounded in the setup. Consequently, the system must provision for a vector length bound that captures all possible plaintext vectors that would be encrypted during the lifetime of the system. Further, the size of ciphertexts and the encryption time, however small the length of the plaintext vector $\boldsymbol{x}$ is, scale with the worst-case vector length bound. Also, in the [AGT21b] construction, each authority can control at most a bounded number of attributes. This could be a bottleneck in certain applications, for instance, a university may introduce a new academic degree program over time which would require its potential to freely expand the attribute list under its control. Moreover, in the MA-ABIPFE system of [AGT21b], new authorities/attributes could not join beyond the upper limit set in the setup. This is clearly a disadvantage for several applications from the point of view of sustainability since it is often impossible to visualize all possible attributes/authorities that can ever come into existence at the time of setting up the system. For instance, new universities may be included in the survey of analyzing mental health of their students, which amplifies the number of authorities/attributes as well as the length of data. Additionally, the MA-ABIPFE scheme of [AGT21b] suffer from the so-called "one-use" restriction, that is, an attribute can appear within an access policy at most a bounded number of times, which clearly limits the class of access policies and negatively impacts efficiency. Lastly, in order to gain confidence in a new cryptographic primitive such as MA-ABIPFE, it is always important to have more and more candidates for that primitive under qualitatively weaker computational assumptions. We thus consider the following open problem:

Open Problem: Is it possible to construct efficient MA-ABIPFE schemes for any expressive class of policies, e.g., LSSS, and avoiding the one-use restriction in prime-order bilinear groups under any (possibly qualitatively weaker) computational assumption such that an arbitrary number of authorities (possibly having an unbounded number of attributes under their control) can join at any point of time and an unbounded length data can be processed?

Our Results

In this paper, we answer the above open problem affirmatively. More precisely, we start by formulating the notion of (decentralized) multi-authority attribute-based *unbounded* IPFE (MA-ABUIPFE) which has all the features discussed above, namely, (a) several independent authorities can control different attributes in the system, (b) authorities can join the system at any time and there is no upper bound on the number of authorities that can ever exist in the system, and (c) unbounded length message and key vectors can be processed, that is, each authority can generate their public and master secret keys without fixing the length of vectors that can be processed with their keys. Next, we construct MA-ABUIPFE supporting LSSS access structures in the significantly faster prime-order bilinear group setting under computational assumptions based in the target group which are known to be qualitatively weaker compared to those based in the source group [BSW13, DKW21b]. The efficiency improvements achieved by our scheme as compared to the only known MA-ABIPFE scheme [AGT21b] is quite significant (see Tables 1.1 and 1.2 for a concrete comparison of the schemes). On a more positive note, we are able to overcome the "one-use restriction", that is, support the appearance of attributes within access policies arbitrarily many times.

We present two MA-ABUIPFE schemes with varying trade-offs between versatility and underlying assumptions.

– **Small-Universe MA-ABUIPFE Scheme**: We construct an MA-ABUIPFE scheme where an authority is allowed to control a single (or a bounded number of) attribute(s), but the number of authorities that could be added to the system is still arbitrary. The construction is proven secure under the decisional bilinear Diffie-Hellman (DBDH) assumption [BF01, SP19] which is a very well-studied computational assumption based in the target groups. Note that the DBDH assumption underlies the security of classical ABE schemes [SW05, GPSW06, Wat11] and has recently been shown to realize MA-ABE [DKW21b]. Our MA-ABUIPFE scheme demonstrates that it is possible to base the security of an even richer functionality on DBDH as well.
– **Large-Universe MA-ABUIPFE Scheme**: We further upgrade our small-universe MA-ABUIPFE scheme to support large attribute universe, that is, where each authority can control exponentially many attributes. We present the security of this construction under a parameterized version of the DBDH assumption which we call the L-DBDH assumption. We justify the validity of this new computational assumption in the generic bilinear group model [Sho97, BBG05] as is done for nearly if not all bilinear group-based computational assumptions used today. Note that, so far, there is no known MA-ABE or even ABE schemes supporting large universe in the literature that is proven secure without parameterized assumption. The efficiency of the proposed large-universe scheme is well comparable to the small-universe one. Thus, our large-universe MA-ABUIPFE (LMA-ABUIPFE) scheme addresses several efficiency and practicality issues towards deploying this primitive in practice.

Since our focus on this paper is on efficiency and practicality, we content with proving the security of our schemes in the static model where the adversary has to declare all its ciphertext, secret key, and authority corruption queries upfront following prior work on MA-ABE with similar motivations [RW15]. However, we would like to mention that while we could not prove our schemes secure against selective adversaries under DBDH or similar target-group-based assumptions, that is, adversaries who must send the challenge ciphertext and authority corruption queries upfront but are allowed to make user secret key queries adaptively afterwards, as considered in [AGT21b], we could not identify any vulnerability in our proposed schemes against such adversaries. Also, just like prior MA-ABE schemes proven secure under standard computational assumptions, we make use of the random oracle model[3].

In order to design our small-universe MA-ABUIPFE, we build on the techniques used in the MA-ABE construction from DBDH by [DKW21b] and the unbounded IPFE construction from DBDH by [SP19]. However, as explained in Section 2 below, a straightforward combination of those techniques does not work. We devise a novel hash-decomposition technique to decompose the evaluation of the hash values, used as randomizers for tying together the different secret keys for the same user, between the encryption and key generation/decryption algorithms and also for handling satisfying and non-satisfying secret key queries of the adversary during the security proof differently. (Please see Section 2 for more details on the hash-decomposition technique.)

Along the way to our small universe MA-ABUIPFE scheme, we also present a single authority ABUIPFE for LSSS access policies in prime-order bilinear groups under the DBDH assumption. Prior to this work, there was no known AB-IPFE scheme even for bounded length vectors that was proven secure under a target group assumption. Thus, the proposed ABUIPFE expands the portfolio of computational assumptions on which this useful primitive can be based on and thereby increasing the confidence in the existence of this primitive in turn. Further, our construction also demonstrates that despite of being a more expressive functionality, MA-ABIPFE is still possible under the same assumption as ABE or MA-ABE. In fact, our AB-IPFE is the first target-group assumption-based FE scheme that goes beyond the "all-or-nothing" paradigm.

[3] Very recently, Waters, Wee, and Wu [WWW22] presented a lattice-based MA-ABE scheme that does not make use of random oracles. However, the scheme relies on a recently introduced complexity assumption called evasive LWE [Wee22] which is a strong knowledge type assumption and is not yet cryptanalyzed in detail.

Table 1.1. Efficiency Comparison of [AGT21b] and Our Scheme with 128-bit Security

Scheme	Group order length (in bits)	$\lvert\mathsf{PK}_t\rvert$/ $\lvert\mathsf{PK}_\theta\rvert$	$\lvert\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\rvert$ $\mathsf{T}(t)=\theta$	$\lvert\mathsf{CT}\rvert$	Encrypt Time	Decrypt Time
Agrawal et al. [AGT21b]	3072	$6054n$	3072	$(n+\ell+2n\ell)3072$	$(n+n\ell)\mathsf{E}_{N,T}+(\ell+n\ell)\mathsf{E}_{N,S}$	$(\ell+1)\mathsf{P}_N+(n+n\ell^2)\mathsf{E}_{N,T}+(\ell+n\ell^2)\mathsf{E}_{N,S}$
MA-ABUIPFE (Section 5)	128	$\lvert\mathsf{PK}_t\rvert=128s_{\max}$	128	$[n+\ell s_{\max}(n+1)]128$	$(n+n\ell)\mathsf{E}_{q,T}+[\ell s_{\max}(n+2)-\ell(n+1)]\mathsf{E}_{q,S}+(2\ell n(s_{\max}-1))\mathsf{P}_q$	$[\ell+n(s_{\max}-1)](\mathsf{P}_q+\mathsf{E}_{q,T})+n\mathsf{E}_{q,S}$
LMA-ABUIPFE (Section 6)	128	$\lvert\mathsf{PK}_\theta\rvert=128s_{\max}$	$128(s_{\max}+1)$	$[n+\ell s_{\max}(n+2)]128$	$(n+n\ell)\mathsf{E}_{q,T}+[\ell s_{\max}(n+3)-\ell(n+1)]\mathsf{E}_{q,S}+(2\ell n(s_{\max}-1))\mathsf{P}_q$	$[\ell+n(s_{\max}-1)](\mathsf{P}_q+\mathsf{E}_{q,T})+\ell s_{\max}\mathsf{P}_q+n\mathsf{E}_{q,S}$

The notations from Table 1.1 are described below:

- $\lvert\mathsf{PK}_t\rvert/\lvert\mathsf{PK}_\theta\rvert$: size of the public key associated to the attribute t or authority θ
- $\lvert\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\rvert$: size of the secret key associated to the tuple $(\mathsf{GID},t,\boldsymbol{u})$
- $\lvert\mathsf{CT}\rvert$: size of the ciphertext
- n: length of vectors; ℓ, $s_{\max}$: number of rows and columns in LSSS matrix respectively
- $\mathsf{E}_{N,S}$, $\mathsf{E}_{q,S}$: exponentiation time in composite and prime order source groups respectively
- $\mathsf{E}_{N,T}$, $\mathsf{E}_{q,T}$: exponentiation time in composite and prime order target groups respectively
- P_N, P_q: time to compute a pairing in composite and prime order groups respectively

Table 1.2. Concrete Efficiency Comparison for 128-bit Security, $n=200$, $\ell=50$, $s_{\max}=30$.

Scheme	$\lvert\mathsf{PK}_\theta\rvert$	$\lvert\mathsf{CT}\rvert$	Encrypt Time	Decrypt Time
Agrawal et al. [AGT21b]	$\approx$ 147.8 KB	$\approx$ 7.42 MB	$\approx$ 143.7 mins.	$\approx$ 4.9 days
MA-ABUIPFE (Section 5)	$\approx$ 0.48 KB	$\approx$ 4.83 MB	$\approx$ 86.7 mins.	$\approx$ 11.03 mins.
LMA-ABUIPFE (Section 6)	$\approx$ 0.48 KB	$\approx$ 4.85 MB	$\approx$ 86.8 mins.	$\approx$ 11.15 mins.

Advantages of Our Schemes Over Agrawal et al. [AGT21b] Beyond Unboundedness: Our MA-ABUIPFE schemes have notable advantages in terms of versatility and performance over the MA-ABIPFE of [AGT21b], named as AGT-FE hereafter beyond the unboundedness property that we achieve in this work. Firstly, the composite-order group-based AGT-FE is significantly slower than our prime-order constructions [Fre10, Gui13] because of the inherent efficiency gains offered by prime-order bilinear groups. Especially, the size of group elements of a composite-order group $\mathbb{G}_N$ is much larger than that of a prime-order group $\mathbb{G}_q$ for the same security level: 3072-bit length of $\mathbb{G}_N$ compared to 128-bit length of $\mathbb{G}_q$ for the 128-bit security level. Moreover, one pairing operation is more than 250 times slower in $\mathbb{G}_N$ compared to its prime-order counterpart. A concrete comparison of efficiency is depicted in Tables 1.1 and 1.2. As we can see, at 128-bit security level, while AGT-FE takes nearly 5 days, our scheme only takes several minutes. We also bring down the ciphertext size by around 40%. Thus our constructions mark a significant progress towards the practical deployment of this primitive. Secondly, the security of AGT-FE is based on source-group-assumptions, precisely, various types of subgroup decision assumptions, which are known to be qualitatively stronger than the target-group-based assumptions [BSW13] such as the DBDH assumption considered in this work. The existing transformations from composite-order group-based systems to analogous prime-order group-based systems [Fre10, Lew12, CGKW18] that could be applied to AGT-FE, technically replaces the subgroup structures by some vector space structures. Consequently, it incurs additional overheads and potential loss in the efficiency to the resulting prime-order system. Further, the translated scheme would still depend on source group assumptions, e.g. the k-linear or its variants.

Thus, our MA-ABUIPFE exhibits a substantial boost with respect to the performance and at the same time it is secure under a weaker assumption. Furthermore, we extend our MA-ABUIPFE to the large universe setting which has the flexibility to include an unbounded number of attributes under different authorities to the system at any point of time.

Static Security: Our Motivation: The static security may not be the dream security model for MA-ABUIPFE. However, in this work, our main motivation is on performance and versatility. Moreover, as we already mentioned above, we could not find any vulnerability of our schemes against stronger adversaries, e.g., selective adversaries as considered in [AGT21b], even though we could not prove it based on the computational assumptions we considered in this paper. Schemes with greater performance and weaker provable security have often found to suit better in practical deployments. Further, weaker security notions have often been a major stepping stone to obtain more advanced security, e.g., adaptive security, for the same primitive. Please note that many primitives like ABE [SW05, GPSW06, Wat11], MA-ABE [RW15, DKW21a, DKW21b, WWW22], IPFE [ABCP15], and MC-IPFE [CSG+18, ABG19], were first built only with selective/static security before being upgraded to adaptive security [ALS16, DKW22, NPP22] based on the same assumptions. Moreover, from a sustainability point of view, it is always important to have a portfolio of candidates for a primitive under various computational assumptions so that if one of the assumptions gets broken, candidates under a different assumption can be deployed. Another motivation for designing a DBDH or related assumption-based scheme is to innovate new techniques that could possibly be translated to the LWE setting, as has previously been done for other FE primitives, e.g., [BF01, ABB10, DKW21a, DKW21b].

Paper Organization: The paper is organized as follows. We provide technical overview of our small and large universe MA-ABUIPFE schemes in Section 2. Important notations, computational assumptions and definitions are given in Section 3. We formalize the notion of small and large universe MA-ABUIPFEs for LSSS in Section 4. In Section 5, we present the construction of small universe MA-ABUIPFE and formally discuss its correctness and security analysis. Next, our LMA-ABUIPFE scheme is described in Section 6 along with its correctness and the security analysis. The small universe single authority ABUIPFE scheme along with its correctness and security analysis is presented in Appendix A. Lastly, we justify the generic security of our newly introduced L-DBDH assumption in Appendix B.

2 Technical Overview

In this technical overview, we focus on discussing the high level technical details of constructing small universe MA-ABUIPFE since this is where most of our technical ideas lie. For extending it to large universe setting, we depend on the technique of Rouselakis and Waters [RW15] which we discuss later in this section. Since our goal is to construct the schemes under target-group-based assumptions, we start with the only existing UIPFE scheme of [SP19] whose security relies on the DBDH assumption. In fact, their UIPFE is designed from the selectively secure (bounded) IPFE of Abdalla et al. [ABCP15] using a hash and pairing mechanism.

2.1 Constructing the Small Universe MA-ABUIPFE

In this overview, we denote by q a prime number and by $[\![x]\!]_i$ an element in a group $\mathbb{G}_i$ for $i \in \{1, 2, T\}$. At a high level, given a public key $[\![\alpha]\!]_1$, the encryption algorithm of [SP19] amplifies entropy by pairing the public key with the outputs of a hash function applied on the indices of the message vectors. More precisely, the ciphertext and secret keys in the [SP19] UIPFE (DP-UIPFE) takes the following forms.

$$\begin{array}{ll} \mathsf{CT}_{\boldsymbol{v}}: & C_0 = [\![r]\!]_1, \quad \{C_i = [\![v_i]\!]_T \cdot e([\![\alpha]\!]_1, r[\![\mathsf{H}(i)]\!]_2)\}_{i \in \mathcal{I}_{\boldsymbol{v}}}; \quad r \leftarrow \mathbb{Z}_q \\ \mathsf{SK}_{\boldsymbol{u}}: & \qquad\qquad -\alpha \prod_{j \in \mathcal{I}_{\boldsymbol{u}}} \mathsf{H}(j)^{u_j} \end{array}$$

where $\mathcal{I}_{\boldsymbol{u}}, \mathcal{I}_{\boldsymbol{v}} \subset \mathbb{N}$ are the index sets of $\boldsymbol{u}, \boldsymbol{v}$ respectively, the hash function H maps the indices to elements in $\mathbb{G}_2$ and $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a prime-order bilinear group. If the index sets are equal, i.e. $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I}$ then one can use the key vector $\boldsymbol{u}$ to extract $[\![\boldsymbol{u} \cdot \boldsymbol{v}]\!]_T$ from the product $\prod_{j \in \mathcal{I}} C_j^{u_j}$ and a single pairing $e(C_0, \mathsf{SK}_{\boldsymbol{u}})$. As a natural first step, we seek to utilize the DP-UIPFE to upgrade an existing MA-ABE to a small universe MA-ABUIPFE scheme.

As the aim is to rely on the target-group-based assumption, we consider the DBDH-based MA-ABE of Datta, Komargodski and Waters (DKW-MA-ABE) [DKW21b] for this upgrade. As a simpler first step, we investigate the primitive in the bounded and small universe setting, that is, the number of authorities and vector lengths are bounded and each authority controls a single attribute.

2.1.1 The First Step: A Bounded MA-ABIPFE Scheme

Let us start by adding the functionality of IPFE on top of DKW-MA-ABE. For each authority t, the public key and master secret key in the DKW-MA-ABE construction are given by $\mathsf{PK}_t = ([\![\alpha_t]\!]_T, [\![y_{t,2}]\!]_1, \ldots, [\![y_{t,s_{\mathsf{max}}}]\!]_1)$ and $\mathsf{MSK}_t = (\alpha_t, y_{t,2}, \ldots, y_{t,s_{\mathsf{max}}})$ where s_{max} is a bound on the maximum number of columns in the LSSS access structure and $\alpha_t, y_{t,2}, \ldots, y_{t,s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q$. In order to construct an MA-ABIPFE scheme from the DKW-MA-ABE, we convert the components of MSK_t from scalars to vectors whose lengths are fixed according to the vector length bound of the system. All the other components are similarly upgraded to either vectors or matrices of *fixed* dimensions. In particular, the resulting MA-ABIPFE derived from DKW-MA-ABE can be described in the following way where $P = (\mathbf{M} = (M_{i,j})_{\ell \times s_{\mathsf{max}}}, \rho : [\ell] \to \mathcal{AU})$ is the LSSS access policy associated with the ciphertexts, $\mathcal{AU}$ is the set of all authorities, and $\boldsymbol{M}_i$ denotes the i-th row of $\mathbf{M}$.

$$\begin{array}{rl} \mathsf{PK}_t: & ([\![\boldsymbol{\alpha}_t]\!]_T, [\![\boldsymbol{y}_{t,2}]\!]_1, \ldots, [\![\boldsymbol{y}_{t,s_{\mathsf{max}}}]\!]_1) \\ \mathsf{MSK}_t: & (\boldsymbol{\alpha}_t, \boldsymbol{y}_{t,2}, \ldots, \boldsymbol{y}_{t,s_{\mathsf{max}}}) \\ \mathsf{CT}_{\boldsymbol{v},P}: & C_0 = [\![\boldsymbol{v} + \boldsymbol{z}]\!]_T, \quad C_{1,i} = [\![\boldsymbol{M}_i \mathbf{B} + r_i \boldsymbol{\alpha}_{\rho(i)}]\!]_T, \\ & C_{2,i} = [\![r_i]\!]_1, \qquad C_{3,i,j} = [\![M_{i,j} \boldsymbol{x}_j + r_i \boldsymbol{y}_{\rho(i),j}]\!]_1 \ \ \forall i \in [\ell], j \in [2, s_{\mathsf{max}}] \\ \mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}: & [\![\boldsymbol{\alpha}_t \cdot \boldsymbol{u}]\!]_2 \cdot \prod_{j=2}^{s_{\mathsf{max}}} \mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j)^{\boldsymbol{y}_{t,j} \cdot \boldsymbol{u}} \end{array}$$

where $\boldsymbol{z} \leftarrow \mathbb{Z}_q^n, r_i \leftarrow \mathbb{Z}_q$ and n represents the length of $\boldsymbol{u}, \boldsymbol{v}$. Further, $\mathbf{B} \in \mathbb{Z}_q^{s_{\mathsf{max}} \times n}$ and $\{\boldsymbol{x}_j \leftarrow \mathbb{Z}_q^n\}_{j \in [2, s_{\mathsf{max}}]}$ are the secret shares of $\boldsymbol{z}$ and $\mathbf{0}$ respectively. Recall that the decryption algorithm of MA-ABIPFE requires a set of secret keys $\{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}_{t \in S}$ for the same user identifier GID and an authorized subset S of attributes featuring in the LSSS access policy associated with the ciphertext in order to decrypt it. Given such a collection of keys , the decryption algorithm gets rid of the masking term from $C_0 \cdot \boldsymbol{u}$ by computing

$$[\![\boldsymbol{u} \cdot \boldsymbol{z}]\!]_T = \prod_{i \in I} \left[\frac{C_{1,i} \cdot \boldsymbol{u} \cdot \prod_{j=2}^{s_{\mathsf{max}}} e\left(\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j), C_{3,i,j} \cdot \boldsymbol{u}\right)}{e\left(\mathsf{SK}_{\mathsf{GID},\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i} \tag{2.1}$$

where I represents the rows of $\mathbf{M}$ associated to S. Note that the Equation (2.1) holds as the decryption algorithm can efficiently find a coefficients $\{w_i \in \mathbb{Z}_q\}_{i \in I}$ satisfying $(1, 0, \ldots, 0) = \sum_{i \in I} w_i \boldsymbol{M}_i$ whenever the attributes linked to the rows in I satisfies the policy $(\mathbf{M}, \rho)$.

The role of the public hash function H is to tie together a set of independently generated secret keys under the same user identifier GID while decrypting. In the security proof, H is treated as a random oracle to ensure that a fresh randomness is produced for each user identity GID that

links together the different secret keys generated for it and it is infeasible for an adversary to mix and match secret keys generated with respect to different global identifiers even if the attributes associated with those secret keys satisfy the access policy associated with the ciphertext.

In fact, the above bounded MA-ABIPFE scheme can be proven secure in the static model under the DBDH assumption. Let us now proceed to transform the bounded scheme into an unbounded one using the idea of DP-UIPFE sketched above. Unfortunately, a straightforward approach does not work. In particular, we face a few difficulties while incorporating the hash and pairing mechanism of [SP19] with the DKW-MA-ABE as we describe below.

2.1.2 Challenges in Expanding Authority Keys on the Fly and Our Approach

The foremost problem arises in vectorizing the components of the authority master secret keys MSK_t. This is because there being no upper bound on the length of vectors, we cannot simply use random vectors of predetermined sizes in the vectorization process. Rather, we must provision for generating the components of the vectors on the fly as needed during encryption/key generation. Similar to the idea of [SP19], we use hash functions modeled as random oracles in order to resolve this issue. More precisely,we proceed as follows: An authority t generates the public/master secret keys as $(\mathsf{PK}_t = ([\![\alpha_t]\!]_T, [\![y_{t,2}]\!]_1, \ldots, [\![y_{t,s_{\max}}]\!]_1), \mathsf{MSK}_t = (\alpha_t, y_{t,2}, \ldots, y_{t,s_{\max}}))$ without knowing the vector lengths where $\alpha, y_{t,2}, \ldots, y_{t,smx}$ are still scalars. To maintain the simplicity of this overview, we assume that the vectors $\boldsymbol{u} = (u_k)_{k\in\mathcal{I}_{\boldsymbol{u}}}$ and $\boldsymbol{v} = (v_k)_{k\in\mathcal{I}_{\boldsymbol{v}}}$ are both associated with the index set $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I} = [n]$ which is unknown to the authority setup. Then the scalar α_t could be vectorized using a hash function H_1 as follows.

$$\text{during encryption}: \quad C_{1,i} = [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T$$
$$\text{where } [\![\vartheta_{i,k}]\!]_T = e(r_i[\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel k \parallel \mathcal{I}))$$
$$\text{during key generation}: \quad \boldsymbol{\alpha}_t \cdot \boldsymbol{u} = \prod_{k=1}^{n} \mathsf{H}_1(t \parallel k \parallel \mathcal{I})^{\alpha_t \cdot u_k}$$

The next step is to vectorize the authority master secret key components $y_{t,j}$ according to the vector lengths. One may hope to apply [SP19] idea to extend $y_{t,j}$ to the same length of the vectors on the fly in a similar way. To see whether it works, let us assume that the hash function H used in the key generation in the above bounded MA-ABIPFE additionally takes an index position and an index set as inputs. That is, let us do the following modification for the key generation of the bounded MA-ABIPFE scheme

$$\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j)^{\boldsymbol{y}_{t,j} \cdot \boldsymbol{u}} \longrightarrow \prod_{k=1}^{n} \mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I})^{y_{t,j} \cdot u_k}$$

Thus, using this idea, it is possible to expand $y_{t,j}$ to a vector $\boldsymbol{y}_{t,j}$ of the same length as the key vector $\boldsymbol{u}$ and eventually enabling an authority to compute the term $\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I})^{\boldsymbol{y}_{t,j} \cdot \boldsymbol{u}}$ while generating keys for an unbounded length vector. Note that, the hash value $\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I})$ has GID and $\boldsymbol{u}$ as inputs. Therefore, this would call for the following modification in the ciphertext computation.

$$C_{3,i,j} = [\![M_{i,j}\boldsymbol{x}_j + \varsigma_{i,j}]\!]_T$$
$$\text{where } [\![\varsigma_{i,j,k}]\!]_T = e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}(\boxed{\mathsf{GID}} \parallel \boxed{\boldsymbol{u}} \parallel j \parallel k \parallel \mathcal{I}))$$

However, such a vector $[\![\boldsymbol{y}_{t,j}]\!]_1$ is not known or rather the k-th element $e([\![y_{t,j}]\!]_1, \mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I}))$ can not be computed during encryption. The main reason is that the global identity GID and the vector $\boldsymbol{u}$ are available when an authority generates a secret key, but the encryption algorithm is oblivious of which GID or $\boldsymbol{u}$ will be used to decrypt the ciphertext. In fact, it is natural that the same ciphertext would be decrypted by several users with different GID and $\boldsymbol{u}$ vectors. Hence, a simple hash and pairing technique similar to DP-UIPFE is not sufficient for a data owner to encrypt unbounded length vectors.

At this point, we devise a correlated "hash-decomposition" mechanism which enables us to compute the value of a hash function by combining the outputs of several hash functions applied on different segments of the input to the original hash function. More precisely, our idea is to define the hash value $\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I})$ by grouping two independently generated hash values as

$$\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I}) = \mathsf{H}_2(j \parallel k \parallel \mathcal{I}) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k) \tag{2.2}$$

where H_2 and H_3 are two new public hash functions generated during global setup. Now, we observe that the first hash value $\mathsf{H}_2(j \parallel k \parallel \mathcal{I})$ in the product can be computed without knowing GID, which in turn enable the encryptor to expand an authority public key component $[\![y_{t,j}]\!]_1$ into a vector $[\![\boldsymbol{y}^{(2)}_{t,j}]\!]_T$ as $[\![y^{(2)}_{t,j,k}]\!]_T = e([\![y_{t,j}]\!]_1, \mathsf{H}_2(j \parallel k \parallel \mathcal{I}))$. Similarly, an authority expands the master secret key component $y_{t,j}$ into vectors $[\![\boldsymbol{y}^{(2)}_{t,j}]\!]_2$ and $[\![\boldsymbol{y}^{(3)}_{t,j}]\!]_2$ as $[\![y^{(2)}_{t,j,k}]\!]_2 = \mathsf{H}_2(j \parallel k \parallel \mathcal{I}))^{y_{t,j}}$ and $[\![y^{(3)}_{t,j,k}]\!]_2 = \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k)^{y_{t,j}}$ respectively while generating a secret key for a vector $\boldsymbol{u}$. However, at this point, it is not immediate how would the vector $[\![\boldsymbol{y}^{(2)}_{t,j}]\!]_T$ be useful for the encryption algorithm.

Next, we carefully look into the decryption equation of the bounded MA-ABIPFE scheme described above (Equation (2.1)) and try to adapt it for the MA-ABUIPFE setting with the modifications we did so far. We note that the pairing operation in the numerator can be rearranged with the hash function H replaced by H_2 as

$$\begin{aligned} e\left(\mathsf{H}_2(j \parallel k \parallel \mathcal{I}), C_{3,i,j} \cdot \boldsymbol{u}\right) &= e(\mathsf{H}_2(j \parallel k \parallel \mathcal{I}), (M_{i,j}\boldsymbol{x}_j + r_i \boldsymbol{y}_{\rho(i),j}) \cdot \boldsymbol{u}) \\ &= e(\mathsf{H}_2(j \parallel k \parallel \mathcal{I}), M_{i,j}\boldsymbol{x}_j \cdot \boldsymbol{u}) \cdot [\![r_i \boldsymbol{y}^{(2)}_{\rho(i),j} \cdot \boldsymbol{u}]\!]_T \end{aligned}$$

Since $\boldsymbol{u}$ is not available during encryption, we only compute the above term without multiplying by $\boldsymbol{u}$ and represent it as a single element

$$C_{3,i,j,k} = e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel k \parallel \mathcal{I})) \cdot [\![r_i y^{(2)}_{\rho(i),j,k}]\!]_T.$$

Therefore, the hash-decomposition mechanism allows the encryptor to simulate the *first* part of the hash value $\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k \parallel \mathcal{I})$ from Equation (2.2) using the hash function H_2. The second part of the hash value still remains to be handled. For this, we generate an additional layer of secret share of zero by sampling $f_2, \ldots, f_{s_{\mathsf{max}}} \in \mathbb{Z}_q$ and introduce the encodings $C_{4,i,j} = [\![M_{i,j}f_j + r_i y_{\rho(i),j}]\!]_1$ for all $i \in [\ell], j \in [2, s_{\mathsf{max}}]$ within the ciphertext. At the time of decryption, $C_{4,i,j}$ will be paired with the term $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k)^{u_k}$. Thus, combining $C_{3,i,j,k}$ and $C_{4,i,j}$ via the hash-decomposition mechanism we are able to distribute the execution of the pairing operation from (Equation (2.1)) among the encryption and decryption algorithms as follows:

$$\begin{aligned} & e\left(\mathsf{H}(\mathsf{GID} \parallel \boldsymbol{u} \parallel j), C_{3,i,j} \cdot \boldsymbol{u}\right) && \text{as in MA-ABIPFE decryption (ref: Equation (2.1))} \\ \longrightarrow & \prod_{k=1}^{n} C_{3,i,j,k} \cdot u_k \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k)^{u_k}) && \text{new decryption strategy for MA-ABUIPFE} \\ = & C^{(3,4)}_{i,j}(\boldsymbol{u}) && \text{(say)} \end{aligned}$$

Equipped with these concepts, we state our final MA-ABUIPFE scheme below by assuming $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I} = [n]$.

$$
\begin{aligned}
\mathsf{PK}_t :\ & (\llbracket \alpha_t \rrbracket_T, \llbracket y_{t,2} \rrbracket_1, \ldots, \llbracket y_{t,s_{\mathsf{max}}} \rrbracket_1) \\
\mathsf{MSK}_t :\ & (\alpha_t, y_{t,2}, \ldots, y_{t,s_{\mathsf{max}}}) \\
\mathsf{CT}_{\boldsymbol{v},P} :\ & C_0 = \llbracket \boldsymbol{v} + \boldsymbol{z} \rrbracket_T, \quad C_{1,i} = \llbracket \boldsymbol{M}_i \mathbf{B} + \boldsymbol{\vartheta}_i \rrbracket_T, \quad C_{2,i} = \llbracket r_i \rrbracket_1, \\
& C_{3,i,j,k} = e(\llbracket M_{i,j} x_{j,k} \rrbracket_1, \mathsf{H}_2(j \parallel k \parallel \mathcal{I})) \cdot \llbracket r_i y^{(2)}_{\rho(i),j,k} \rrbracket_T, \\
& C_{4,i,j} = \llbracket M_{i,j} f_j + r_i y_{\rho(i),j} \rrbracket_1, \quad \forall i \in [\ell],\ j \in [2, s_{\mathsf{max}}],\ k \in [n] \\
\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} :\ & \prod_{k=1}^{n} \mathsf{H}_1(t \parallel k \parallel \mathcal{I})^{\alpha_t \cdot u_k} \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\llbracket y^{(2)}_{t,j,k} \rrbracket_2 \cdot \llbracket y^{(3)}_{t,j,k} \rrbracket_2)^{u_k}
\end{aligned}
$$

The components $\boldsymbol{\vartheta}_i, y^{(2)}_{t,j,k}, y^{(3)}_{t,j,k}$ are defined as above. The decryption follows by canceling the masking term from $C_0 \cdot \boldsymbol{u}$ using a similar computation like in Equation (2.1) executed as

$$
\llbracket \boldsymbol{u} \cdot \boldsymbol{z} \rrbracket_T = \prod_{i \in I} \left[\frac{C_{1,i} \bullet \boldsymbol{u} \cdot \prod_{j=2}^{s_{\mathsf{max}}} C^{(3,4)}_{i,j}(\boldsymbol{u})}{e\left(\mathsf{SK}_{\mathsf{GID},\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i} \tag{2.3}
$$

We next look into the security of the proposed construction. Here again, we face several challenges while adapting the security proof of [SP19, DKW21b] into our setting.

2.1.3 Challenges in the Security Analysis and Our Approach

The main difference between the MA-ABE and MA-ABUIPFE security model is in the secret key queries made by the adversary. This is because MA-ABUIPFE is more like an FE scheme and the adversary is entitled to ask for secret keys that would decrypt the challenge ciphertext which is in contrast to any MA-ABE scheme where only non-authorized keys are released. On the other hand, proving security of MA-ABUIPFE is more technically challenging compared to the (bounded) MA-ABIPFE (like AGT-FE [AGT21b]) as an authorized key which always leads to a successful decryption in case of MA-ABIPFE, may not be eligible for decrypting a ciphertext of MA-ABUIPFE. The index set associated with the authorized key must match to the index set of the encrypted vector for successful decryption in MA-ABUIPFE. In other words, the adversary should be restricted to infer any information about the encrypted message vector from the authorized keys whose index sets are not equal to the index set of the message vector. Moreover, AGT-FE is proven secure under subgroup decision assumptions which are source group assumptions while our target is to prove security under DBDH which is a target group assumption, thus the dual system encryption technique [Wat09] used for the security proof of AGT-FE does not work in our case. Hence, we design a different proof strategy that works coherently with the hash-decomposition mechanism and for target group assumptions in the prime-order bilinear group.

We prove the security of our MA-ABUIPFE in the static model similar to the DKW-MA-ABE. The adversary is asked to submit all it's queries including the challenge message vectors $\boldsymbol{v}_0, \boldsymbol{v}_1$ with a common index set $\mathcal{I}^*$ and an associated challenge access structure $(\mathbf{M}, \rho)$. Recall that the adversary can also corrupt or even maliciously generate some of the authorities indicated by a set $\mathcal{C}$ of corrupted authorities or attributes. Let us consider a DBDH instance $(\llbracket a \rrbracket_1, \llbracket b \rrbracket_2, \llbracket c \rrbracket_1, \llbracket \tau \rrbracket_T)$ where τ is either abc or random. In the first step, we use the information-theoretic *partitioning* lemma, the so-called "zero-out" lemma [RW15, Lemma 1], to isolate and ignore the set of rows of $\mathbf{M}$ that correspond to the corrupted authorities throughout the analysis. In particular, the lemma allows us to replace the LSSS matrix $\mathbf{M}$ with an updated simpler matrix $\mathbf{M}'$ such that a subset of columns, say $C_{\mathbf{M}'}$, of $\mathbf{M}'$ can be set to zero that are related to the corrupted authorities. Next, we follow the proof techniques of [ABCP15, SP19] and sample a basis $\widetilde{S} = \{(\boldsymbol{v}_0 - \boldsymbol{v}_1), \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ of $\mathbb{Z}_q^n$ where n denotes the size of $\mathcal{I}^*$ to represent key vectors $\boldsymbol{u}$ whose lengths are equal to n. However, answering the hash and secret key queries require a careful treatment while embedding the DBDH challenge instance. The role of the hash function of DKW-MA-ABE was limited to simulating the

non-authorized keys of a fixed length. However, in our case, we need to deal with both authorized and unauthorized keys and here again, our hash-decomposition mechanism plays a crucial role. Moreover, a key can be non-authorized with respect to the index set or the associated policy, or both.

Let S be the set of attributes queried under a user identifier GID as a part of secret key queries such that S contains at least an attribute involved in the challenge policy. The main idea of simulating secret keys of DKW-MA-ABE was to sample a special vector $\boldsymbol{d} \in \mathbb{Z}_q^{s_{\max}}$ such that the inner product of $\boldsymbol{d}$ with $\boldsymbol{M}'_i$ is zero for all $i \in \rho^{-1}(S \cup \mathcal{C})$ and to set the hash values as

$$\mathsf{H}(\mathsf{GID} \parallel j) = (g_2^b)^{d_j} \cdot g_2^{h_j},\ \forall j \in C_{\mathbf{M}'},\ \text{and uniform otherwise.} \tag{2.4}$$

This, in fact, enables in simulating the secret keys using the properties of $\boldsymbol{d}$ and by embedding the matrix $\mathbf{M}'$ into the public keys of authorities linked to the challenge policy. Unfortunately, we observe that such encoding of hash values is not compatible with our hash-decomposition mechanism. Firstly, the hash function H_2 does not take a GID as input and hence it is not possible to encode the hash values depending on a vector like $\boldsymbol{d}$ which is sampled according to an unauthorized set of attributes $(S \cup \mathcal{C})$ under a given global identity. In our case, H_2 should generate a good amount of entropy for indices of key vectors irrespective of any global identity. This would restrict an adversary to gain any illegitimate information about the encrypted message from any secret key where the associated index set does not match with $\mathcal{I}^*$ even though the attributes associated to the key satisfy the challenge policy. Secondly, H_3 takes a GID as it's input along with a key vector, a column number and an index set. The role of H_3 is to make a secret key generated under a given GID useless to the adversary whenever the associated attributes does not satisfy the challenge policy.

In the static security model the simulator knows all the secret key queries in advance. We exploit this fact to prepare encodings for the hash values keeping in mind their roles in the security experiment. Our idea is to sample all possible $\{\boldsymbol{d}_\phi\}_\phi$ vectors corresponding to the sets $\{S_\phi \cup \mathcal{C}\}_\phi$ such that $S_\phi \cup \mathcal{C}$ constitutes an unauthorized subset of row of $\mathbf{M}$ and use the information of $\{\boldsymbol{d}_\phi\}_\phi$ in the encodings of the hash functions. More precisely, we use an *add and subtract* technique to set the hash values as follows

$$\mathsf{H}_2(j \parallel k \parallel \mathcal{I}^*) = (g_2^b)^{\sum_\phi d_{\phi,j}} \cdot g_2^{h_{2,j}},\ \forall j \in C_{\mathbf{M}'},\ \text{and uniform otherwise.}$$
$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_{\phi'} \parallel j \parallel k) = (g_2^b)^{\sum_{\phi \neq \phi'} -d_{\phi,j}} \cdot g_2^{h_{3,j}},\ \forall j \in C_{\mathbf{M}'},\ \text{and uniform otherwise.}$$

Now, we multiply the above hash encodings while simulating non-authorized secret key queries and obtain a hash encoding similar to Equation (2.4).

$$\mathsf{H}_2(j \parallel k \parallel \mathcal{I}^*) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_{\phi'} \parallel j \parallel k) = (g_2^b)^{d_{\phi',j}} \cdot g_2^{h_{2,j}+h_{3,j}}\ \forall j \in C_{\mathbf{M}'}.$$

For simplicity of this section, we have ignored a few additional elements in the above encodings that connect the hash values with the H_1 encodings which actually facilitates in using the fact that $\boldsymbol{d}_\phi \cdot \boldsymbol{M}'_i = 0$ for all $i \in \rho^{-1}(S_\phi \cup \mathcal{C})$ for non-authorized keys such that $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$. Lastly, when simulating authorized secret keys we use the basis $\widetilde{S}$ to obtain a vector $\boldsymbol{\eta}$ satisfying $\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi = 0$ with the help of the admissibility condition $\boldsymbol{u}_\phi \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) = 0$ for all keys leading to a successful decryption of the challenge ciphertext. The full security analysis can be found in Section 5.3.

2.2 Constructing the Large Universe MA-ABUIPFE

We recall that in the large universe setting each authority is allowed to control exponentially many attributes. We upgrade our small universe scheme to a large universe MA-ABUIPFE (LMA-ABUIPFE)

by extending the techniques presented in [RW15] from encrypting a fixed length message to encrypting an unbounded length vector in the context of MA-ABUIPFE. To support exponentially many attributes, we use an additional hash function R which maps arbitrary attributes to elements of $\mathbb{G}_2$. We replace the map ρ of the LSSS access structure $(\mathbf{M}, \rho)$ by decomposition of two mappings T and δ, that is $\rho(i) = \mathsf{T}(\delta(i)) = \theta$ where δ labels row numbers i of the LSSS access matrix to some attributes $\delta(i)$ and T assigns the attributes $\delta(i)$ to its respective authorities denoted by θ. Our LMA-ABUIPFE is described as follows.

$$
\begin{aligned}
\mathsf{PK}_\theta :\ & (\llbracket \alpha_\theta \rrbracket_T, \llbracket y_{\theta,2} \rrbracket_1, \ldots, \llbracket y_{\theta,s_{\mathsf{max}}} \rrbracket_1) \\
\mathsf{MSK}_\theta :\ & (\alpha_\theta, y_{\theta,2}, \ldots, y_{\theta,s_{\mathsf{max}}}) \\
\mathsf{CT}_{\boldsymbol{v},P} :\ & \begin{array}{c} C_0 = \llbracket \boldsymbol{v} + \boldsymbol{z} \rrbracket_T, \quad C_{1,i} = \llbracket \boldsymbol{M}_i \mathbf{B} + \boldsymbol{\vartheta}_i \rrbracket_T, \quad C_{2,i} = \llbracket r_i \rrbracket, \\ C_{3,i,j,k} = e(\llbracket M_{i,j} x_{j,k} \rrbracket_1, \mathsf{H}_2(j \parallel k \parallel \mathcal{I})) \cdot \llbracket r_i y^{(2)}_{\rho(i),j,k} \rrbracket_T, \\ C_{4,i,j} = \llbracket M_{i,j} f_j + r_i y_{\rho(i),j} \rrbracket_1, \ \ C_{5,i,j} = \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}) \\ \forall i \in [\ell],\ j \in [2, s_{\mathsf{max}}],\ k \in [n] \end{array} \\
\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} :\ & \prod_{k=1}^{n} \mathsf{H}_1(t \parallel k \parallel \mathcal{I})^{\alpha_\theta \cdot u_k} \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\llbracket y^{(2)}_{\theta,j,k} \rrbracket_2 \cdot \llbracket y^{(3)}_{\theta,j,k} \rrbracket_2)^{u_k} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I})^{\tau_j}, \\
& \mathsf{Z}_{t,j} = \llbracket \tau_j \rrbracket_1, \ \ \forall j \in [s_{\mathsf{max}}]
\end{aligned}
$$

The components $\boldsymbol{\vartheta}_i, \boldsymbol{y}^{(2)}_{\theta,j}, \boldsymbol{y}^{(3)}_{\theta,j}$ are defined similarly as in our MA-ABUIPFE scheme.

$$
\begin{aligned}
\llbracket \vartheta_{i,k} \rrbracket_T &= e(r_i \llbracket \alpha_{\rho(i)} \rrbracket_1, \mathsf{H}_1(\rho(i) \parallel k \parallel \mathcal{I}_{\boldsymbol{v}})), \\
\llbracket y^{(2)}_{\theta,j,k} \rrbracket_T &= e(\llbracket y_{\theta,j} \rrbracket_1, \mathsf{H}_2(j \parallel k \parallel \mathcal{I})), \ \ \llbracket y^{(2)}_{\theta,j,k} \rrbracket_2 = \mathsf{H}_2(j \parallel k \parallel \mathcal{I})^{y_{\theta,j}}, \\
\llbracket y^{(3)}_{\theta,j,k} \rrbracket_2 &= \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel k)^{y_{\theta,j}}, \ \ \forall k \in [n].
\end{aligned}
$$

The decryption procedure is similar to our MA-ABUIPFE scheme. We consider static security of LMA-ABUIPFE and model the hash functions as random oracles. However, it may not be possible to base security on the plain DBDH assumption. Following the same notations that we used to sketch the proof technique of our MA-ABUIPFE, we discuss the main reason which prevent using the DBDH assumption as before. The R-values related to the authorities in the challenge policy in our proposed LMA-ABUIPFE scheme described above are roughly set as $\mathsf{R}(t \parallel j \parallel \mathcal{I}^*) = g_2^{\zeta_{t,j}} g_2^{a M'_{i,j}}$, where $\zeta_{t,j}$ is a random $\mathbb{Z}_q$-element and $M'_{i,j}$ is the (i,j)-th entry of the updated LSSS matrix $\mathbf{M}'$ in the challenge policy. On the other hand, the randomness r_i used in the encryption[4] are set as $r_i = c$. Hence, the reduction requires the group element g_2^{ac} in order to simulate the components $C_{5,i,j}$ of the challenge ciphertext. However, the DBDH assumption does not make it possible to make g^{ac} available to an adversary.

Thus, for basing the security, we look into the parameterized versions of the DBDH assumptions. Unlike [RW15] where they consider a much more complex parameterized assumption, a primary motivation of our security reduction is to depend on a simpler parameterized assumption that is as close as possible to the plain DBDH assumption. More specifically, [RW15] consider an *exponent* type assumption where each instance consists of at least $O(L^3_{\mathsf{max}})$ group elements and $L_{\mathsf{max}} \geq \max\{\ell, s_{\mathsf{max}}\}$, where ℓ, s_{max} is the number of rows and columns of the challenge LSSS access matrix respectively. Consequently, the reduction becomes more involved and complex. In contrast, we prove the security of LMA-ABUIPFE based on the newly introduced L-DBDH assumption where each instance has $O(L^2)$ group elements with $L \geq \ell$. We show that the L-DBDH assumption is generically secure using the techniques of [BBG05, RW15]. Although incomparable with the assumption used in [RW15], it seems that our L-DBDH assumption is weaker as it contains

[4] The ciphertext is re-randomized to ensure the distribution of its components is unharmed.

fewer elements. Therefore, our LMA-ABUIPFE improves upon the previous results of [RW15] even without considering the enhanced functionality of UIPFE.

There are some other technical hurdles in the security reduction that does not directly allow using the *program and cancel* technique similar to [RW15] while simulating secret key queries. This is due to the fact that we are handling unbounded length messages and using a hash-decomposition mechanism on top of large universe paradigm. In contrast to the small universe scheme, an authority in a queried secret key of LMA-ABUIPFE may be present in the challenge policy but none of their attributes are linked to it. We use our *add and subtract* technique which enables the reduction to combine the decomposed hash values into a single hash value that eventually produces an adequate amount of randomness preventing the leakage of unwanted information about the underlying message vector from such secret keys.

On the other hand, if the authorities as well as some of their controlled attributes are present in the challenge policy but the associated secret key is unauthorized then we observe that the program and cancel technique of [RW15] is not sufficient to handle an adversary of LMA-ABUIPFE given the fact that it can query for secret keys corresponding to vectors of arbitrary lengths. In order to make these secret keys useless for an adversary irrespective of the associated lengths of vectors, we delicately program the hash queries that enables the reduction to procreate additional entropy via an interplay between the *program and cancel* technique of [RW15] and *add and subtract* mechanism of ours at the time of simulating such unauthorized secret keys. Although the high-level proof technique is inspired from [RW15], the technical obstacles mentioned above prevent applying their approach straightforwardly into our setting. As a whole, we carefully embed the L-DBDH instance into the adversary's queries by extending the [RW15] technique in the context of amplifying entropy for supporting computation over unbounded length vectors and at the same time making it compatible for hash-decomposition mechanism used in our scheme. We present a detailed security analysis in Section 6.3.

3 Preliminaries

In this section, we present the notations used in this paper and the new L-DBDH assumption we introduce.

3.1 Notations

We will denote the underlying security parameter by λ throughout the paper. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is said to be a negligible function of λ, if for every $c \in \mathbb{N}$, there exists a $\lambda_c \in \mathbb{N}$ such that $\forall \lambda > \lambda_c$, $\mathsf{negl}(\lambda) < \lambda^{-c}$. We denote the set of positive integers $\{1, \ldots, n\}$ as $[n]$. We denote $\emptyset$ as the empty set. We use the abbreviation PPT for probabilistic polynomial-time. For a set X, we write $x \leftarrow X$ to denote that x is sampled according to the uniform distribution over the elements of X. Also for any set X, we denote by $|X|$ and 2^X the cardinality and the power set of the set X respectively. We use bold lower case letters, such as $\boldsymbol{v}$, to denote vectors and upper-case, such as $\mathbf{M}$, for matrices. We assume all vectors, by default, are row vectors. The i^{th} row of a matrix is denoted by $\boldsymbol{M}_i$ and analogously for a set of row indices I, we denote $\mathbf{M}_I$ for the sub-matrix of $\mathbf{M}$ that consists of the rows $\mathbf{M}_i, \forall i \in I$. By $\mathsf{rowspan}(\mathbf{M})$, we denote the linear span of the rows of a matrix $\mathbf{M}$.

For an integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. We represent $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$. The set of matrices of size $m \times n$ with elements in $\mathbb{Z}_q$ is denoted by $\mathbb{Z}_q^{m \times n}$. The operation $(\cdot)^\top$ denotes the transpose of vectors/matrices. Let $\boldsymbol{u} = (u_i)_{i \in \mathcal{I}_{\boldsymbol{u}}} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{u}}|}$, $\boldsymbol{v} = (v_i)_{i \in \mathcal{I}_{\boldsymbol{v}}} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{v}}|}$ where $\mathcal{I}_{\boldsymbol{u}}$ and $\mathcal{I}_{\boldsymbol{v}}$ are the associated index sets, then the inner product between the vectors is denoted as $\boldsymbol{v} \cdot \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{v} = \sum_{i \in \mathcal{I}} u_i v_i \in \mathbb{Z}_q$ whenever $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I}$.

3.2 Bilinear Groups and Complexity Assumptions

Assume a bilinear group generator algorithm $\mathcal{G}$ that takes as input 1^λ and outputs a tuple $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, e)$, where $\mathbb{G}_1, \mathbb{G}_2$ are the source groups and $\mathbb{G}_T$ is the target group of the same prime order $q = q(\lambda)$ with generators g_1, g_2 and g_T respectively. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies the following properties:

- *Bilinearity*: $\forall a, b \in \mathbb{Z}_q, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- *Non-degeneracy*: $e(g_1, g_2) = g_T$ generates $\mathbb{G}_T$.

For any $a \in \mathbb{Z}_q$, we define $[\![a]\!]_i := g_i^a \in \mathbb{G}_i$ where $i \in \{1, 2, T\}$. The notation is analogously extended for vectors $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ as $[\![\boldsymbol{a}]\!]_i \in \mathbb{G}_i^n$ or matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ as $[\![\mathbf{A}]\!]_i \in \mathbb{G}_i^{n \times m}$ by entry wise exponentiation. For $[\![\boldsymbol{a}]\!]_i \in \mathbb{G}_i^n$ and $\boldsymbol{b} \in \mathbb{Z}_q^n$, we denote $\boldsymbol{b} \cdot [\![\boldsymbol{a}]\!]_i$ by the group element $[\![\boldsymbol{a} \cdot \boldsymbol{b}]\!]_i \in \mathbb{G}_i$ which is efficiently computable given the vectors $[\![\boldsymbol{a}]\!]_i$ and $\boldsymbol{b}$.

We formally define the DBDH assumption and a parameterized version of it, we call L-DBDH which would underlie of security of our small and large universe MA-ABUIPFE schemes respectively.

Assumption 3.1 (Decisional Bilinear Diffie-Hellman (DBDH) [BLS01, SP19]) *For a security parameter $\lambda \in \mathbb{N}$, let $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda)$ be a bilinear group and let $a, b, c \leftarrow \mathbb{Z}_q$. The* DBDH *assumption states that for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that for any security parameter $\lambda \in \mathbb{N}$, given the distribution $(\mathsf{G}, [\![a]\!]_1, [\![c]\!]_1, [\![a]\!]_2, [\![b]\!]_2, [\![\tau]\!]_T)$, $\mathcal{A}$ has advantage*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DBDH}}(\lambda) = \left|\Pr\left[1 \leftarrow \mathcal{A}\left(1^\lambda, \mathcal{D}, [\![abc]\!]_T\right)\right] - \Pr\left[1 \leftarrow \mathcal{A}\left(1^\lambda, \mathcal{D}, [\![\tau]\!]_T\right)\right]\right| \leq \mathsf{negl}(\lambda),$$

Assumption 3.2 (L-Decisional Bilinear Diffie-Hellman (L-DBDH)) *Let $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda)$ be a bilinear group and let $a, b, c, \mu_1, \ldots, \mu_L \leftarrow \mathbb{Z}_q$. The L-*DBDH *assumption states that for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that for any security parameter $\lambda \in \mathbb{N}$, given the distribution*

$$\left(\mathsf{G}, \begin{pmatrix}[\![b]\!]_1, [\![c]\!]_1, \\ [\![a]\!]_2, [\![b]\!]_2\end{pmatrix}, \left\{\begin{matrix}[\![a\mu_i]\!]_1, [\![c/\mu_i]\!]_1, \\ [\![a\mu_i]\!]_2\end{matrix}\right\}_{i \in [L]}, \left\{\begin{matrix}[\![c\mu_\iota/\mu_i]\!]_1, [\![ac\mu_\iota/\mu_i]\!]_1, \\ [\![ac\mu_\iota/\mu_i]\!]_2\end{matrix}\right\}_{\substack{i, \iota \in [L], \\ i \neq \iota}}, [\![\tau]\!]_T\right)$$

$\mathcal{A}$ has advantage

$$\mathsf{Adv}_{\mathcal{A}}^{L\text{-}\mathsf{DBDH}}(\lambda) = \left|\Pr\left[1 \leftarrow \mathcal{A}\left(1^\lambda, \mathcal{D}, [\![abc]\!]_T\right)\right] - \Pr\left[1 \leftarrow \mathcal{A}\left(1^\lambda, \mathcal{D}, [\![\tau]\!]_T\right)\right]\right| \leq \mathsf{negl}(\lambda).$$

We establish the generic security of L-DBDH in generic bilinear pairing group model in Appendix B.

3.3 Access Structures and Linear Secret Sharing Schemes

In this subsection, we present the formal definitions of access structures and linear secret-sharing schemes. This subsection is taken verbatim from [DKW21a, Subsection 3.2].

Definition 3.1 (Access Structures) *Let $\mathcal{AU}$ be the attribute universe. An access structure on $\mathcal{AU}$ is a collection $\mathbb{A} \subseteq 2^{\mathcal{AU}} \setminus \emptyset$ of non-empty sets of attributes. The sets in $\mathbb{A}$ are called the* authorized *sets and the sets not in $\mathbb{A}$ are called the* unauthorized *sets. An access structure is called* monotone *if $\forall B, C \in 2^{\mathcal{AU}}$ if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.*

Definition 3.2 (Linear Secret Sharing Schemes (LSSS)) *Let $q = q(\lambda)$ be a prime and $\mathcal{AU}$ the attribute universe. A secret sharing scheme Π with domain of secrets $\mathbb{Z}_q$ for a monotone access structure $\mathbb{A}$ over $\mathcal{AU}$, a.k.a. a monotone secret sharing scheme, is a randomized algorithm that on input a secret $z \in \mathbb{Z}_q$ outputs $|\mathcal{AU}|$ shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_{|\mathcal{AU}|}$ such that for any set $S \in \mathbb{A}$ the shares $\{\mathsf{sh}_i\}_{i \in S}$ determine z and other sets of shares are independent of z (as random variables). A secret-sharing scheme Π realizing monotone access structures on $\mathcal{AU}$ is linear over $\mathbb{Z}_q$ if*

1. *The shares of a secret $z \in \mathbb{Z}_q$ for each attribute in $\mathcal{AU}$ form a vector over $\mathbb{Z}_q$.*
2. *For each monotone access structure $\mathbb{A}$ on $\mathcal{AU}$, there exists a matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times s}$, called the share-generating matrix, and a function $\rho \colon [\ell] \to \mathcal{AU}$, that labels the rows of $\mathbf{M}$ with attributes from $\mathcal{AU}$ which satisfy the following: During the generation of the shares, we consider the vector $\boldsymbol{v} = (z, r_2, ..., r_s)$, where $r_2, \ldots, r_s \leftarrow \mathbb{Z}_q$. Then the vector of ℓ shares of the secret z according to Π is given by $\boldsymbol{\mu} = \mathbf{M}\boldsymbol{v}^\top \in \mathbb{Z}_q^{\ell \times 1}$, where for all $j \in [\ell]$ the share μ_j "belongs" to the attribute $\rho(j)$. We will be referring to the pair $(\mathbf{M}, \rho)$ as the* LSSS *policy of the access structure $\mathbb{A}$.*

The correctness and security of a monotone LSSS *are formalized in the following: Let S (resp. S') denote an authorized (resp. unauthorized) set of attributes according to some monotone access structure $\mathbb{A}$ and let I (resp. I') be the set of rows of the share generating matrix $\mathbf{M}$ of the* LSSS *policy pair $(\mathbf{M}, \rho)$ associated with $\mathbb{A}$ whose labels are in S (resp. S'). For correctness, there exist constants $\{w_i\}_{i \in I}$ in $\mathbb{Z}_q$ such that for any valid shares $\{\boldsymbol{\mu}_i = (\mathbf{M}\boldsymbol{v}^\top)_i\}_{i \in I}$ of a secret $z \in \mathbb{Z}_q$ according to Π, it is true that $\sum_{i \in I} w_i \boldsymbol{\mu}_i = z$ (equivalently, $\sum_{i \in I} w_i \mathbf{M}_i = (1, \overbrace{0, \ldots, 0}^{s-1})$, where $\mathbf{M}_i$ is the ith row of $\mathbf{M}$). For soundness, there are no such w_i's, as above. Additionally, we have that $\exists \boldsymbol{d} \in \mathbb{Z}_q^{1 \times s}$, such that its first component $d_1 = 1$ and $\mathbf{M}_i \cdot \boldsymbol{d} = 0, \forall i \in I'$.*

Remark 3.1 (NC^1 and Monotone LSSS) *Consider an access structure $\mathbb{A}$ described by an* NC^1 *circuit. There is a folklore transformation that can convert this circuit by a Boolean formula of logarithmic depth that consists of (fan-in 2)* AND*,* OR*, and (fan-in 1)* NOT *gates. We can further push the* NOT *gates to the leaves using De Morgan laws, and assume that internal nodes only constitute of* OR *and* AND *gates and leaves are labeled either by attributes or their negations. In other words, we can represent any* NC^1 *policy over a set of attributes into one described by a monotone Boolean formula of logarithmic depth over the same attributes and their negations. Lewko and Waters [LW11] presented a monotone* LSSS *for access structures described by monotone Boolean formulas. This implies that any* NC^1 *access policy can be captured by a monotone* LSSS*. Therefore, in this paper, we will only focus on designing an* MA-ABIPFE *schemes for monotone* LSSS *similar to the* MA-ABE *scheme of Datta et al. [DKW21b].*

We will use the following information theoretic property of LSSS access policies in the security proof of our MA-ABUIPFE or LMA-ABUIPFE scheme. This lemma first appeared in [RW15, Lemma 1]. Recently, Datta, Komargodski, and Waters [DKW21a] observed a gap in the proof of [RW15] and presented a corrected proof; for details see [DKW21a, Section 4.3]. The security reduction of the MA-ABE scheme of Datta, Komargodski, and Waters [DKW21a] crucially utilize this lemma to isolate an unauthorized set of rows of the challenge LSSS matrix submitted by the adversary and essentially ignore it throughout the security reduction. Like [RW15, DKW21a, DKW21b], in our case as well, the rows of the challenge LSSS matrix corresponding to the corrupt authorities will constitute the unauthorized set in the application of the lemma.

Lemma 3.1 *Let $(\mathbf{M}, \rho)$ be an* LSSS *access policy, where $\mathbf{M} \in \mathbb{Z}_q^{\ell \times s}$. Let $\mathcal{C} \subset [\ell]$ be a non-authorized subset of row indices of $\mathbf{M}$. Let $c \in \mathbb{N}$ be the dimension of the subspace spanned by the rows of $\mathbf{M}$ corresponding to indices in $\mathcal{C}$. Then, there exists an access policy $(\mathbf{M}', \rho)$ such that the following holds:*

– *The matrix* $\mathbf{M}' = (M'_{i,j})_{\ell \times s} \in \mathbb{Z}_q^{\ell \times s}$ *satisfies* $M'_{i,j} = 0$ *for all* $(i,j) \in \mathcal{C} \times [s-c]$.
– *For any subset* $\mathcal{S} \subset [\ell]$*, if the rows of* $\mathbf{M}$ *having indices in* $\mathcal{S}$ *are linearly independent, then so are the rows of* $\mathbf{M}'$ *with indices in* $\mathcal{S}$.
– *The distribution of shares* $\{\mu_x\}_{x \in [\ell]}$ *sharing a secret* $z \in \mathbb{Z}_q$ *generated with the matrix* $\mathbf{M}$ *is the same as the distribution of the shares* $\{\mu'_x\}_{x \in [\ell]}$ *sharing the same secret* z *generated with the matrix* $\mathbf{M}'$.

4 Definition of Decentralized (Large Universe) MA-ABUIPFE for LSSS

A large universe decentralized multi-authority attribute-based inner-product functional encryption (LMA-ABUIPFE) scheme $\mathsf{LMA\text{-}ABUIPFE} = (\mathsf{GlobalSetup}, \mathsf{LocalSetup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ for access structures captured by linear secret sharing schemes (LSSS) over some finite field $\mathbb{Z}_q$ with $q = q(\lambda)$ and inner product value space $\mathcal{U}$ consists of five algorithms with the following syntax. We denote by $\mathcal{AU}$ the authority universe and by $\mathcal{GID}$ the universe of users' global identifiers in the system. The attribute universe is denoted as $\mathsf{U}_{\mathsf{att}}$ which may be arbitrary. Further, an authority $\theta \in \mathcal{AU}$ may have any arbitrary number of attributes from $\mathsf{U}_{\mathsf{att}}$ under its control. Following [RW15], we assume a publicly computable function $\mathsf{T} : \mathsf{U}_{\mathsf{att}} \to \mathcal{AU}$ that maps each attribute $t \in \mathsf{U}_{\mathsf{att}}$ to a unique authority $\theta = T(t)$. The algorithms proceed as follows:

$\mathsf{GlobalSetup}(1^\lambda, s_{\mathsf{max}})$: It is the global setup algorithm which on input the security parameter λ and a maximum width s_{max} of the LSSS matrix, and outputs the global public parameters GP. We assume that GP includes the descriptions of $\mathcal{AU}$ and $\mathcal{GID}$.

$\mathsf{LocalSetup}(\mathsf{GP}, \theta)$: The authority $\theta \in \mathcal{AU}$ runs the local setup algorithm during its initialization with the global parameters GP and generates its public parameters and a master secret key pair $(\mathsf{PK}_\theta, \mathsf{MSK}_\theta)$.

$\mathsf{KeyGen}(\mathsf{GP}, \mathsf{GID}, \mathsf{MSK}_\theta, t, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$: The key generation algorithm takes input the global parameter GP, a user's global identifier $\mathsf{GID} \in \mathcal{GID}$, a master secret key MSK_θ for authority θ controlling an attribute $t \in \mathsf{U}_{\mathsf{att}}$, and a vector $\boldsymbol{u} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{u}}|}$ with an associated index set $\mathcal{I}_{\boldsymbol{u}}$. It outputs a secret key $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}$ which contains $(\boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$.

$\mathsf{Encrypt}(\mathsf{GP}, (\mathbf{M}, \delta), \{\mathsf{PK}_\theta\}_\theta, \boldsymbol{v}, \mathcal{I}_{\boldsymbol{v}})$: The encryption algorithm takes input the global parameter GP, an LSSS access structure $(\mathbf{M}, \delta)$ where $\mathbf{M}$ is a matrix over $\mathbb{Z}_q$ and δ is a row-labeling function that assigns to each row of $\mathbf{M}$ an attribute in $\mathsf{U}_{\mathsf{att}}$. We define the function $\rho : [\ell] \to \mathcal{AU}$ as $\rho(\cdot) := \mathsf{T}(\delta(\cdot))$ which maps row indices of $\mathbf{M}$ to authorities $\theta \in \mathcal{AU}$. Accordingly, the encryption algorithm further takes a set $\{\mathsf{PK}_\theta\}_\theta$ of public keys for all the authorities in the range of ρ, and a message vector $\boldsymbol{v} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{v}}|}$ with an associated index set $\mathcal{I}_{\boldsymbol{v}}$. It outputs a ciphertext CT. We assume that CT implicitly contains the description of $(\mathbf{M}, \delta)$ and $\mathcal{I}_{\boldsymbol{v}}$.

$\mathsf{Decrypt}(\mathsf{GP}, \mathsf{GID}, \mathsf{CT}, \{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}_t)$: The decryption algorithm takes in the global parameters GP, a ciphertext CT generated with respect to some LSSS access policy $(\mathbf{M}, \delta)$ and an index set $\mathcal{I}$ associated to the message, and a collection of keys $\{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}_t$ corresponding to user ID-attribute pairs $(\mathsf{GID}, S \subseteq \mathsf{U}_{\mathsf{att}})$ and a key vector $(\boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$ possessed by a user with global identifier GID. It outputs a message ζ when the collection of attributes associated with the secret keys $\{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}_t$ satisfies the LSSS access policy $(\mathbf{M}, \delta)$, i.e., when the vector $(1, 0, \ldots, 0)$ belongs to the linear span of those rows of $\mathbf{M}$ which are mapped by δ to the set of attributes in S that corresponds to the secret keys $\{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}_{t \in S}$ possessed by the user with global identifier GID. Otherwise, decryption returns $\perp$.

Correctness: An LMA-ABUIPFE scheme for LSSS-realizable access structures and inner product message space $\mathcal{U}$ is said to be correct if for every $\lambda \in \mathbb{N}$, every message vector $\boldsymbol{v} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{v}}|}$, key vector $\boldsymbol{u} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{u}}|}$ such that $\mathcal{I} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I}_{\boldsymbol{u}}$, and $\mathsf{GID} \in \mathcal{GID}$, every LSSS access policy $(\mathbf{M}, \delta)$, and

every subset of authorities $S \subseteq \mathsf{U_{att}}$ controlling attributes which satisfy the access structure it holds that

$$\Pr\left[\Gamma = \boldsymbol{v}\cdot\boldsymbol{u} \;\middle|\; \begin{array}{c} \mathsf{GP} \leftarrow \mathsf{GlobalSetup}(1^\lambda, 1^n), \\ (\mathsf{PK}_\theta, \mathsf{MSK}_\theta) \leftarrow \mathsf{LocalSetup}(\mathsf{GP}, \theta), \\ \mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{GP}, \mathsf{GID}, \mathsf{MSK}_\theta, t, \boldsymbol{u}), \\ \mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{GP}, (\mathbf{M}, \delta), \{\mathsf{PK}_\theta\}_\theta, \boldsymbol{v}), \\ \Gamma = \mathsf{Decrypt}(\mathsf{GP}, \mathsf{CT}, \{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}_{t\in S}) \end{array}\right] = 1.$$

Static Security: In this paper, we consider static security for LMA-ABUIPFE formalized by the following game between a challenger and an adversary. The static security model is adapted from [RW15], defined for MA-ABE, to the context of LMA-ABUIPFE. We emphasize that unlike MA-ABE, our static security model allows the adversary to ask for secret keys which are capable of decrypting the challenge ciphertext.

Global Setup: The challenger runs $\mathsf{GlobalSetup}(1^\lambda, s_{\mathsf{max}})$ to get and send the global public parameters GP to the attacker.

Adversary's Queries: The adversary sends the following queries:

1. A list $\mathcal{C} \subset \mathcal{AU}$ of corrupt authorities and their respective public parameters $\{\mathsf{PK}_\theta\}_{\theta\in\mathcal{C}}$, which it might have created in a malicious way.
2. A set $\mathcal{N} \subset \mathcal{AU}$ of non-corrupt authorities, i.e., $\mathcal{C}\cap\mathcal{N} = \emptyset$, for which the adversary requests the public keys.
3. A set $\mathcal{Q} = \{(\mathsf{GID}, S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})\}$ of secret key queries with $\mathsf{GID} \in \mathcal{GID}, S \subseteq \mathsf{U_{att}}$ such that $\mathsf{T}(S)\cap\mathcal{C} = \emptyset$, $\boldsymbol{u} \in \mathbb{Z}^{|\mathcal{I}_{\boldsymbol{u}}|}$ and $\mathcal{I}_{\boldsymbol{u}} \subset \mathbb{Z}$ where GIDs are distinct in each of these tuples.
4. Two message vectors $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathbb{Z}_q^{|\mathcal{I}^*|}$ having the same index set $\mathcal{I}^*$, and a challenge LSSS access policy $(\mathbf{M}, \delta)$ with $\mathbf{M} = (M_{i,j})_{\ell\times s_{\mathsf{max}}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell\times s_{\mathsf{max}}}$, $\delta : [\ell] \to \mathsf{U_{att}}$ and satisfying the constraint that for each $(\mathsf{GID}, S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}}) \in \mathcal{Q}$, either $S \cup \bigcup_{\theta\in\mathcal{C}} \mathsf{T}^{-1}(\theta) \subseteq [\ell]$ constitutes an unauthorized subset of rows of the access matrix $\mathbf{M}$ or the secret key vector $\boldsymbol{u}$ satisfies the relation $(\boldsymbol{v}_0 - \boldsymbol{v}_1)\cdot\boldsymbol{u} = 0$ whenever $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}^*$. Note that the set $\bigcup_{\theta\in\mathcal{C}} \mathsf{T}^{-1}(\theta)$ contains the attributes belonging to the corrupt authorities.

Challenger's Replies: The challenger flips a random coin $\beta \leftarrow \{0, 1\}$ and replies with the following:

1. The public keys $\mathsf{PK}_\theta \leftarrow \mathsf{LocalSetup}(\mathsf{GP}, \theta)$ for all $\theta \in \mathcal{N}$.
2. The secret keys $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{GP}, \mathsf{GID}, \mathsf{MSK}_\theta, t, \boldsymbol{u})$ for all $(\mathsf{GID}, S, \boldsymbol{u}) \in \mathcal{Q}, t \in S$.
3. The challenge ciphertext $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{GP}, (\mathbf{M}, \delta), \{\mathsf{PK}_\theta\}_{\theta\in\mathcal{C}\cup\mathcal{N}}, \boldsymbol{v}_\beta)$.

Guess: The adversary outputs a guess β' for β.

The advantage of the adversary $\mathcal{A}$ is $\mathsf{Adv}^{\mathsf{LMA\text{-}ABUIPFE}}_{\mathcal{A},\mathsf{SS\text{-}CPA}}(\lambda) \triangleq |\Pr[\beta = \beta'] - 1/2|$.

Definition 4.1 (Static Security for LMA-ABUIPFE for LSSS) *An* LMA-ABUIPFE *scheme for* LSSS*-realizable access structures satisfies static security if for any* PPT *adversary* $\mathcal{A}$ *there exists* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, we have* $\mathsf{Adv}^{\mathsf{LMA\text{-}ABUIPFE}}_{\mathcal{A},\mathsf{SS\text{-}CPA}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Remark 4.1 (Static Security in the Random Oracle Model.) *Similar to [RW15,DKW21a,DKW21b], we additionally consider the aforementioned notion of selective security with static corruption in the ROM. In this context, we assume a global hash function* H *published as part of the global public parameters and accessible by all the parties in the system. In the security proof, we will model* H *as a random oracle programmed by the challenger. In the security game, therefore, we let the adversary* $\mathcal{A}$ *submit a collection of* H*-oracle queries to the challenger immediately after seeing the global public parameters, along with all the other queries it makes in the static security game as described above.*

Remark 4.2 (Small Universe MA-ABUIPFE.) *The above definition of* LMA-ABUIPFE *captures the large universe scenario where one authority can control multiple attributes. We can similarly*

define a small universe MA-ABUIPFE *or simply* MA-ABUIPFE *by restricting each authority to control only a single attribute [RW15]. Hence, we would use the words* "authority" *and* "attribute" *interchangeably in the case of* MA-ABUIPFE. *There are a few syntactic and semantic changes in the above definition when adapted for the small universe setting:*

1. *There is a bijection between the attribute universe* $\mathsf{U}_{\mathsf{att}}$ *and the authority universe* $\mathcal{AU}$.
2. $\mathsf{LocalSetup}(\mathsf{GP}, t)$ *outputs* $(\mathsf{PK}_t, \mathsf{MSK}_t)$ *for an authority/attribute* $t \in \mathcal{AU}$.
3. $\mathsf{KeyGen}(\mathsf{GP}, \mathsf{GID}, \mathsf{MSK}_t, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$ *outputs* $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}$.
4. *For an* LSSS *access structure* (M, δ), *we have* $\rho(\cdot) = \delta(\cdot)$ *is an injective map.*
5. *The changes in the security definition follow accordingly. Due to space constraints, we state them directly in the proof of our small universe scheme in Section 5.3.*

5 The Proposed Small Universe MA-ABUIPFE from DBDH

In this section, we describe the formal construction and proof for our MA-ABUIPFE scheme. The construction is in prime-order groups and uses a hash functions that will be modelled as a random oracle in the security proof.

5.1 The Construction

$\mathsf{GlobalSetup}(1^\lambda, s_{\mathsf{max}})$: The global setup algorithm takes input the security parameter λ, the maximum width of an LSSS matrix supported by the scheme $s_{\mathsf{max}} = s_{\mathsf{max}}(\lambda)$ and the vector length n in unary. It generates $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, e)$. Consider the hash functions $\mathsf{H}_1 : \mathsf{U}_{\mathsf{att}} \times \mathbb{Z} \times \mathbb{Z}^* \to \mathbb{G}_2$, $\mathsf{H}_2 : [s_{\mathsf{max}}] \times \mathbb{Z} \times \mathbb{Z}^* \to \mathbb{G}_2$, $\mathsf{H}_3 : \mathcal{GID} \times \mathbb{Z}^* \times [s_{\mathsf{max}}] \to \mathbb{G}_2$. It outputs a global parameter $\mathsf{GP} = (\mathsf{G}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$.

$\mathsf{LocalSetup}(\mathsf{GP}, t)$: The authority setup algorithm takes as input GP and an authority index/attribute $t \in \mathcal{AU}$. It samples vectors $\alpha_t, y_{t,2}, \ldots, y_{t,s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q$ and outputs

$$\mathsf{PK} = \left(\{[\![\alpha_t]\!]_1, \{[\![y_{t,j}]\!]_1\}_{j\in\{2,\ldots,s_{\mathsf{max}}\}}\}_{t\in\mathsf{U}_{\mathsf{att}}}\right), \quad \mathsf{MSK} = \{\{\alpha_t, \{y_{t,j}\}_{j\in\{2,\ldots,s_{\mathsf{max}}\}}\}_{t\in\mathsf{U}_{\mathsf{att}}}\}$$

$\mathsf{KeyGen}(\mathsf{GP}, \mathsf{GID}, \mathsf{MSK}_t, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$: The key generation algorithm takes input GP, the user's global identifier GID, the authority's secret key MSK_t and a vector $\boldsymbol{u} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{u}}|}$. It proceeds as follows

1. Parse $\mathcal{I}_{\boldsymbol{u}} = \{\iota_1, \ldots, \iota_n\}$ and $\boldsymbol{u} = (u_{\iota_1}, \ldots, u_{\iota_n})$.
2. Compute
$$\mathsf{SK}_{t,\boldsymbol{u}} = \prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}})^{\alpha_t u_{\iota_k}} \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}}) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{y_{t,j} u_{\iota_k}}.$$
3. Output $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \mathsf{SK}_{t,\boldsymbol{u}}, \mathcal{I}_{\boldsymbol{u}})$ as the secret key.

$\mathsf{Encrypt}(\mathsf{GP}, (\mathrm{M}, \rho), \{\mathsf{PK}_t\}, \boldsymbol{v}, \mathcal{I}_{\boldsymbol{v}})$: The encryption algorithm takes input the global parameter GP, an LSSS access structure (M, ρ) where $\mathrm{M} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and $\rho : [\ell] \to \mathcal{AU}$, a set $\{\mathsf{PK}_t\}$ of public keys for all the authorities in the range of ρ, and a message vector $\boldsymbol{v} \in \mathbb{Z}_q^m$. The function maps the row indices of M to authorities or attributes. We assume ρ is an injective function, that is, an authority/attribute is associated with at most one row of M. The algorithm proceeds as follows:

1. Parse $\mathcal{I}_{\boldsymbol{v}} = \{\iota_1, \ldots, \iota_m\}$ and $\boldsymbol{v} = (v_{\iota_1}, \ldots, v_{\iota_m})$.
2. Sample $\{r_i \leftarrow \mathbb{Z}_q\}_{i\in[\ell]}$ and $\boldsymbol{f} = (f_2, \ldots, f_{s_{\mathsf{max}}}) \leftarrow \mathbb{Z}_q^{s_{\mathsf{max}}-1}$.
3. Sample $\boldsymbol{z}, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{s_{\mathsf{max}}}, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q^m$.
4. Set the matrix $\mathrm{B} = \left[\boldsymbol{z}, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{s_{\mathsf{max}}}\right]^\top_{s_{\mathsf{max}} \times m}$.
5. Compute $\vartheta_{i,k} = e(r_i [\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}}))$ and set $\boldsymbol{\vartheta}_i := (\vartheta_{i,1}, \ldots, \vartheta_{i,m})$.

6. Compute the following terms:
$$C_0 = [\![\boldsymbol{v} + \boldsymbol{z}]\!]_T, \quad C_{1,i} = [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T, \quad C_{2,i} = [\![r_i]\!]_1,$$
$$C_{3,i,j,k} = e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})) \cdot e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})),$$
$$C_{4,i,j} = [\![M_{i,j}f_j + y_{\rho(i),j}r_i]\!]_1$$
for all $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [m]$.
7. Output the ciphertext
$\mathsf{CT} = \left((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i\in[\ell],j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[m]}, \mathcal{I}_{\boldsymbol{v}}\right)$.

Decrypt(GP, GID, CT, $\{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}$): The decryption algorithm takes input the public key PK, a secret key $\mathsf{SK}_{S,\boldsymbol{u}}$ for an attribute set $S \subseteq \mathsf{U}_{\mathsf{att}}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a ciphertext CT for an access structure $(\mathbf{M}, \rho)$ with $\mathbf{M} \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and an injective map $\rho : [\ell] \to \mathsf{U}_{\mathsf{att}}$.

Parse $\mathsf{SK}_{\mathsf{GID},S,\boldsymbol{u}} = \left(\mathsf{GID}, \boldsymbol{u}, \{\mathsf{SK}_{\rho(i),\boldsymbol{u}}\}_{\rho(i)\in S}, \mathcal{I}_{\boldsymbol{u}}\right)$, where $i \in [\ell]$ and $\mathsf{CT} = ((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i\in[\ell],j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[m]}, \mathcal{I}_{\boldsymbol{v}})$. Denote $I = \{i | \rho(i) \in S\} \subseteq [\ell]$. If $(1, 0, \ldots, 0)$ is not in the span of $\mathbf{M}_I$ (i.e., $\mathbf{M}$ restricted to the set of rows from I) or $\mathcal{I}_{\boldsymbol{u}} \neq \mathcal{I}_{\boldsymbol{v}}$ decryption returns $\perp$. Else, when S satisfies $(\mathbf{M}, \rho)$, the algorithm finds $\{w_i \in \mathbb{Z}_q\}_{i\in I}$ such that $(1, 0, \ldots, 0) = \sum_{i\in I} w_i\mathbf{M}_i$. It then computes $[\![\Gamma]\!]_T = C_0 \bullet \boldsymbol{u} \cdot [\![\mu]\!]_T$ where $[\![\mu]\!]_T$ is given by

$$\left(\prod_{i\in I}\left[\frac{C_{1,i} \bullet \boldsymbol{u} \cdot \prod_{j=2}^{s_{\mathsf{max}}}\prod_{k=1}^{n} u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}})}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}\right]^{w_i}\right)^{-1}$$

and outputs $\log_{g_T}([\![\Gamma]\!]_T)$.

5.2 Correctness

Consider a secret key $\mathsf{SK}_{\mathsf{GID},S,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \{\mathsf{SK}_{t,\boldsymbol{u}}\}_{t\in S}, \mathcal{I}_{\boldsymbol{u}})$ consisting of a set of attributes satisfying the LSSS access structure $(\mathbf{M}, \rho)$ associated with a ciphertext $\mathsf{CT} = ((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i\in[\ell],j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[m]}, \mathcal{I}_{\boldsymbol{v}})$ such that $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I}$. In particular, the vector $(1, 0, \ldots, 0) \in \mathsf{rowspan}(\mathbf{M}_I)$ corresponding to the set of indices $I = \{i \in I | \rho(i) = t \in S\}$.
For each $i \in I$, we have the following:

$$e(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}) = \prod_{k=1}^{n} e(g_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}))^{r_i\alpha_{\rho(i)}u_{\iota_k}} \cdot \prod_{j=2}^{s_{\mathsf{max}}}\prod_{k=1}^{n}(e(g_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I})) \cdot e(g_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)))^{r_i y_{\rho(i),j} u_{\iota_k}}$$

For $i \in I, j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$,

$$u_{\iota_k} C_{3,i,j,k} = e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{u_{\iota_k}} \cdot e(g_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{r_i y_{\rho(i),j} u_{\iota_k}}$$

For $i \in I, j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$,

$$e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}}) = e([\![M_{i,j}f_j]\!]_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{u_{\iota_k}} \cdot e(g_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{r_i y_{\rho(i),j} u_{\iota_k}}$$

Finally, for each $i \in I$, we have $C_{1,i} = [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T$ and so

$$\frac{C_{1,i} \cdot \boldsymbol{u} \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}$$

$$= [\![\boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u}]\!]_T \prod_{k=1}^{n} e(g_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}))^{r_i \alpha_{\rho(i)} u_{\iota_k}} \cdot$$

$$\frac{\prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}$$

$$= [\![\boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u}]\!]_T \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} e([\![M_{i,j} x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{u_{\iota_k}} \cdot$$

$$\prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} e([\![M_{i,j} f_j]\!]_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{u_{\iota_k}}$$

Since $\mathsf{SK}_{S,\boldsymbol{u}}$ corresponds to a set of qualified authorities, $\exists \{w_i \in \mathbb{Z}_q\}_{i \in I}$ such that $\sum_{i \in I} w_i \boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u} = (1, 0, \ldots, 0)\mathbf{B} \cdot \boldsymbol{u} = \boldsymbol{z} \cdot \boldsymbol{u}$ and it holds that $\sum_{i \in I} w_i M_{i,j} = 0, \forall j \in \{2, \ldots, s_{\max}\}$. Hence, we have

$$\prod_{i \in I} \left[\frac{C_{1,i} \cdot \boldsymbol{u} \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i}$$

$$= [\![\sum_{i \in I} w_i \boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u}]\!]_T = [\![\boldsymbol{z} \cdot \boldsymbol{u}]\!]_T$$

Finally, the message is recovered as $\log_{g_T}([\![\Gamma]\!]_T)$ where

$$[\![\Gamma]\!]_T = (C_0 \cdot \boldsymbol{u})/[\![\boldsymbol{z} \cdot \boldsymbol{u}]\!]_T = [\![\boldsymbol{v} \cdot \boldsymbol{u} + \boldsymbol{z} \cdot \boldsymbol{u}]\!]_T / [\![\boldsymbol{z} \cdot \boldsymbol{u}]\!]_T = [\![\boldsymbol{v} \cdot \boldsymbol{u}]\!]_T.$$

5.3 Security Analysis

Theorem 5.1 *If the* DBDH *assumption holds, then all* PPT *adversaries have a negligible advantage in breaking the static security of the proposed small universe* MA-ABUIPFE *scheme in the random oracle model.*

Proof. We prove this theorem by showing that if there is any PPT adversary $\mathcal{A}$ who breaks the static security of MA-ABUIPFE then there is a PPT adversary $\mathcal{B}$ who solves the DBDH problem with a non-negligible advantage. Suppose, $\mathcal{B}$ gets an instance $(\mathsf{G}, [\![a]\!]_1, [\![c]\!]_1, [\![a]\!]_2, [\![b]\!]_2, [\![\tau]\!]_T)$ of the DBDH problem where $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda)$ is a group description, the elements $a, b, c \leftarrow \mathbb{Z}_q$ are random integers, and the element $\tau \in \mathbb{Z}_q$ is either abc or a random element of $\mathbb{Z}_q$. The algorithm $\mathcal{B}$ works as follows: On input λ, $\mathcal{A}$ outputs $s_{\max}, \mathsf{U}_{\mathsf{att}}$ and queries the following.

Attacker's Queries: Upon initialization, the adversary $\mathcal{A}$ sends the following to $\mathcal{B}$:

(a) A list $\mathcal{C} \subset \mathcal{AU}$ of corrupt authorities and their respective public key

$$\{\mathsf{PK}_t = (\mathrm{Y}_{t,1}, \mathrm{Y}_{t,2}, \ldots, \mathrm{Y}_{t,s_{\max}})\}_{t \in \mathcal{C}},$$

where $\mathrm{Y}_{t,1}, \mathrm{Y}_{t,2}, \ldots, \mathrm{Y}_{t,s_{\max}} \in \mathbb{G}_1$ for all $t \in \mathcal{C}$.

(b) A set $\mathcal{N} \subset \mathcal{AU}$ of non-corrupt authorities, i.e., $\mathcal{C} \cap \mathcal{N} = \emptyset$, for which $\mathcal{A}$ requests the public keys.

(c) A collection of hash queries $\mathcal{H}_1 = \{(t, \iota_k, \mathcal{I}) : t \in \mathsf{U}_{\mathsf{att}}, \iota_k \in \mathbb{Z}, \mathcal{I} \subset \mathbb{N}\}$, $\mathcal{H}_2 = \{(j, \iota_k, \mathcal{I}) : j \in \{2, \ldots, s_{\mathsf{max}}\}, \iota_k \in \mathbb{Z}, \mathcal{I} \subset \mathbb{N}\}$ and $\mathcal{H}_3 = \{(\mathsf{GID}, \boldsymbol{u}, j, \iota_k) : \mathsf{GID} \in \mathcal{GID}, \boldsymbol{u} \in \mathbb{Z}^*, j \in \{2, \ldots, s_{\mathsf{max}}\}, \iota_k \in \mathbb{Z}\}$.

(d) A set $\mathcal{Q} = \{(\mathsf{GID}, S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})\}$ of secret key queries with $\mathsf{GID} \in \mathcal{GID}, S \subseteq \mathsf{U}_{\mathsf{att}}$, $\boldsymbol{u} \in \mathbb{Z}^{|\mathcal{I}_{\boldsymbol{u}}|}$ and $\mathcal{I}_{\boldsymbol{u}} \subset \mathbb{Z}$.

(e) Two message vectors $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathbb{Z}_q^n$ having the same index set $\mathcal{I}^*$, and a challenge LSSS access policy $(\mathbf{M}, \rho)$ with $\mathbf{M} = (M_{i,j})_{\ell \times s_{\mathsf{max}}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and $\rho : [\ell] \to \mathcal{C} \cup \mathcal{N}$ injective and satisfying the constraint that for each $(S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}}) \in \mathcal{Q}_{\boldsymbol{u}}$, either $\rho^{-1}(\mathcal{C} \cup S) \subseteq [\ell]$ constitutes an unauthorized subset of rows of the access matrix $\mathbf{M}$ or the secret key vector $\boldsymbol{u}$ satisfies the relation $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u} = 0$ whenever $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}^*$.

Before answering $\mathcal{A}$'s queries, the adversary $\mathcal{B}$ substitute the secret sharing matrix $\mathbf{M}$ with the matrix $\mathbf{M}'$ from Lemma 3.1 computed using $\rho^{-1}(\mathcal{C})$ as the unauthorized subset of rows. Lemma 3.1 guarantees the fact that if $\mathcal{B}$ uses $\mathbf{M}'$ instead of $\mathbf{M}$ in the simulation, the view of $\mathcal{A}$ in the simulated game is information theoretically the same as if $\mathcal{B}$ would have used the original matrix $\mathbf{M}$. Furthermore, Lemma 3.1 implies that if we assume the subspace spanned by $\mathbf{M}_{\rho^{-1}(\mathcal{C})}$ has dimension $\widetilde{c}$, then so is the dimension of the subspace spanned by $\mathbf{M}'_{\rho^{-1}(\mathcal{C})}$ and $M'_{i,j} = 0$ for all $(i, j) \in \rho^{-1}(\mathcal{C}) \times [s_{\mathsf{max}} - \widetilde{c}]$. $\mathcal{B}$ now proceeds to answer the queries of $\mathcal{A}$. Denote $\widehat{s_{\mathsf{max}}} = s_{\mathsf{max}} - \widetilde{c}$, where $\widetilde{c}$ is the dimension of the sequence spanned by the rows of $\mathbf{M}_{\rho^{-1}(\mathcal{C})}$, the latter being the rows of $\mathbf{M}$ controlled by corrupted authorities, $\mathcal{C}$.

Note that $\mathcal{I}^*$ can be any subset of $\mathbb{Z}$ and w.l.o.g one can consider $\mathcal{I}^* = [n]$[5] for some $n \in \mathbb{N}$. Inspired by the proof techniques of prior works [ABCP15, SP19], the reduction first compute a basis of $(\boldsymbol{v}_0 - \boldsymbol{v}_1)^\perp$ as $\{\widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_{n-1}\}$. Then the set $\widetilde{\mathcal{S}} = \{\boldsymbol{v}_0 - \boldsymbol{v}_1, \widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_{n-1}\}$ form a basis of $\mathbb{Z}_q^n$. For any vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, if we represent it as the linear combination of the vectors in $\widetilde{\mathcal{S}}$ as

$$\boldsymbol{u} = \zeta \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) + \sum_{k=1}^{n-1} \zeta_k \widetilde{\boldsymbol{b}}_k, \quad \text{for some } \zeta, \zeta_k \in \mathbb{Z}_q$$

then $\zeta = 0$ whenever it holds that $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u} = 0$. Let $\boldsymbol{e}_k$ be the k-th vector in the standard basis of $\mathbb{Z}_q^n$. We write $\boldsymbol{e}_i$ for each $i \in [n]$ as

$$\boldsymbol{e}_i = \eta_i \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) + \sum_{k=1}^{n-1} \lambda_{i,k} \widetilde{\boldsymbol{b}}_k \quad \text{for some } \eta, \lambda_{i,k} \in \mathbb{Z}_q.$$

Generating Public Key: There are two cases to consider:

1. **Case 1** — $t \in \mathcal{N} \setminus \rho([\ell])$ (i.e., attribute t is absent in the challenge policy $(\mathbf{M}, \rho)$ but it belongs to a non-corrupt authority) — In this case, $\mathcal{B}$ executes the Setup algorithm according to the real experiment. It samples $\alpha_t, y_{t,2}, \ldots, y_{t,s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q$ by itself, and computes the public key component corresponding to attribute t as $([\![\alpha_t]\!]_1, [\![y_{t,2}]\!]_1, \ldots, [\![y_{t,s_{\mathsf{max}}}]\!]_1)$.
2. **Case 2** — $t \in \rho([\ell]) \setminus \mathcal{C}$ (i.e., attribute t appears in the challenge policy $(\mathbf{M}, \rho)$ and it does not belong to a corrupt authority) — In this case, $\mathcal{B}$ samples $\alpha'_t, y'_{t,2}, \ldots, y'_{t,s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q$ and implicitly sets $\alpha_t = \alpha'_t + a \cdot M'_{\rho^{-1}(t),1}$ and $y_{t,j} = y'_{t,j} + aM'_{\rho^{-1}(t),j}$ for $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$ and $y_{t,j} = y'_{t,j}$ for $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$(these are well-defined as ρ is injective), and sets the public key elements w.r.t. attribute t as $([\![\alpha_t]\!]_1, [\![y_{t,2}]\!]_1, \ldots, [\![y_{t,s_{\mathsf{max}}}]\!]_1)$ where the elements $[\![\alpha_t]\!]_1$ and $[\![y_{t,j}]\!]_1$ for $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$ are computed as follows:

[5] In particular, we consider a map $\gamma : \mathcal{I}^* \to [n]$ and use $\gamma(k) = \iota_k$ throughout the security analysis.

$$[\![\alpha_t]\!]_1 = [\![\alpha'_t]\!]_1 \cdot M'_{\rho^{-1}(t),1}[\![a]\!]_1,\ [\![y_{t,j}]\!]_1 = [\![y'_{t,j}]\!]_1 \cdot M'_{\rho^{-1}(t),j}[\![a]\!]_1 \tag{5.1}$$

for all $j \in [2, \widehat{s_{\mathsf{max}}}]$. Note that, α_t and $\{y_{t,j}\}_{j\in\{2,\ldots,s_{\mathsf{max}}\}}$ are distributed uniformly over $\mathbb{Z}_q$ and hence each of these elements of the public key is properly distributed.

Answering Hash Queries:

1. **H_1 queries.** If $(\iota_k \in \mathcal{I}^* \wedge \mathcal{I} = \mathcal{I}^*)$, then sample uniformly random elements $h_{1,\widehat{k}}, h_{1,t,\iota_k}$ from $\mathbb{Z}_q$ and set

$$\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}) = (g_2^b)^{\eta_k} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{1,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{1,t,\iota_k}}. \tag{5.2}$$

   Otherwise, if $(\iota_k \notin \mathcal{I}^* \vee \mathcal{I} \neq \mathcal{I}^*)$, then output a random $\mathbb{G}_2$ element, i.e., sample uniformly random element h'_{1,t,ι_k} from $\mathbb{Z}_q$ and set $\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}) = g_2^{h'_{1,t,\iota_k}}$. The reduction stores the hash queries for future use.
2. **H_2 queries.** If $(\iota_k \in \mathcal{I}^* \wedge \mathcal{I} = \mathcal{I}^*)$, then sample uniformly random elements $h_{2,\widehat{k}}, h_{2,j,\iota_k}$ for $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$ (in Eq. 5.3) and elements h'_{2,j,ι_k} for $j \in \{\widehat{s_{\mathsf{max}}}+1, \ldots, s_{\mathsf{max}}\}$ from $\mathbb{Z}_q$ (in Eq. 5.4) and set

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,j,\iota_k}} \tag{5.3}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = g_2^{h'_{2,j,\iota_k}} \tag{5.4}$$

   where Q denotes the total number of *non-accepting* key queries $\{(S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi})\}_{\phi\in[Q]}$ made by the adversary in the case where $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$ but the attributes in S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \rho)$. Note that, for such secret key queries, there exists a vector $\boldsymbol{d}_\phi = (d_{\phi,1}, \ldots, d_{\phi,s_{\mathsf{max}}}) \in \mathbb{Z}_q^{s_{\mathsf{max}}}$ such that $d_{\phi,1} = 1$ and the inner product $\boldsymbol{M}'_i \cdot \boldsymbol{d}_\phi = 0$ for all $i \in \rho^{-1}(\mathcal{C} \cup S_\phi)$, where $\boldsymbol{M}'_i$ denotes the i-th row of $\mathbf{M}'$. Additionally, the set of rows $\mathcal{R} = \{\boldsymbol{M}'_i \in \mathbb{Z}_q^{s_{\mathsf{max}}} : i \in \rho^{-1}(\mathcal{C})\}$ has dimension c and $M'_{i,j} = 0$ for all $(i,j) \in \rho^{-1}(\mathcal{C}) \times [\widehat{s_{\mathsf{max}}}]$. Therefore, $\mathcal{R}$ spans the entire subspace $\mathbb{V} = \left\{(\overbrace{0, \ldots, 0}^{\widehat{s_{\mathsf{max}}}}, \boldsymbol{\nu}) : \boldsymbol{\nu} \in \mathbb{Z}_q^c\right\}$. Thus, it follows that $\boldsymbol{d}_\phi$ is orthogonal to any of the vectors

$$\left\{(\overbrace{0, \ldots, 0}^{\widehat{s_{\mathsf{max}}}}, \overbrace{0, \ldots, 0}^{j-1}, 1, \overbrace{0, \ldots, 0}^{c-j})\right\}_{j\in\{\widehat{s_{\mathsf{max}}}+1,\ldots,s_{\mathsf{max}}\}}.$$

   In other words, $d_{\phi,j} = 0$ for all $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$. Combining the above two facts, we have $(\boldsymbol{M}'_i|_{[\widehat{s_{\mathsf{max}}}]}) \cdot (\boldsymbol{d}_\phi|_{[\widehat{s_{\mathsf{max}}}]}) = 0$ for all $i \in \rho^{-1}(S_\phi)$, where for a vector $\boldsymbol{x}$, $\boldsymbol{x}|_X$ denotes a vector formed by taking the entries of $\boldsymbol{x}$ having indices in the set $X \in \mathbb{N}$. For simplicity of notation, let us denote $\boldsymbol{M}'_i \star \boldsymbol{d}_\phi = (\boldsymbol{M}'_i|_{[\widehat{s_{\mathsf{max}}}]}) \cdot (\boldsymbol{d}_\phi|_{[\widehat{s_{\mathsf{max}}}]})$ for $i \in \rho^{-1}(S_\phi)$.
   Otherwise, if $(\iota_k \notin \mathcal{I}^* \vee \mathcal{I} \neq \mathcal{I}^*)$, then output a random $\mathbb{G}_2$ element, i.e., sample uniformly random element h''_{2,t,ι_k} from $\mathbb{Z}_q$ and set $\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = g_2^{h''_{2,t,\iota_k}}$. The reduction stores the hash queries for future use.

3. **H_3 queries.** If $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $S_\phi \cap \rho([\ell]) \neq \emptyset$ and $\rho^{-1}(S_\phi \cup \mathcal{C})$ constitutes an unauthorized subset of the rows of $\mathbf{M}$ then sample h_{3,j,ι_k} for $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$ (in Eq. 5.5) and elements h'_{3,j,ι_k} for $j \in \{\widehat{s_{\mathsf{max}}}+1, \ldots, s_{\mathsf{max}}\}$ from $\mathbb{Z}_q$ (in Eq. 5.6) and set

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = (g_2^b)^{\eta_k \sum_{\phi' \in [Q] \setminus \{\phi\}} -d_{\phi',j}} \cdot g_2^{h_{3,j,\iota_k}} \tag{5.5}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h'_{3,j,\iota_k}} \tag{5.6}$$

for all $\iota_k \in \mathcal{I}_{\boldsymbol{u}_\phi}$ such that $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$ and $\boldsymbol{d}_\phi$ is as defined above.
If $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $S_\phi \cap \rho([\ell]) \neq \emptyset$ and $\mathcal{I}_{\boldsymbol{u}_\phi} \neq \mathcal{I}^*$ then sample h''_{3,j,ι_k} uniformly at random from $\mathbb{Z}_q$ and set $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h''_{3,j,\iota_k}}$.
On the other hand, if $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $S_\phi \cap \rho([\ell]) \neq \emptyset$ and $\rho^{-1}(S_\phi \cup \mathcal{C})$ constitutes an authorized subset of the rows of $\mathbf{M}$ then sample $h'''_{3,j,\iota_k} \leftarrow \mathbb{Z}_q$ and set $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h'''_{3,j,\iota_k}}$. The reduction stores the hash queries for future use.
For all other cases, the reduction simple outputs a uniformly random element from $\mathbb{G}_2$ to answer the hash query $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k)$.

Generating Secret Keys: For any $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$, $\mathcal{B}$ returns a secret key $\mathsf{SK}_{\mathsf{GID},S_\phi,\boldsymbol{u}_\phi} = \big(\mathsf{GID}, \boldsymbol{u}_\phi, \{\mathsf{SK}_{t,\boldsymbol{u}_\phi}\}_{t \in S_\phi}, \mathcal{I}_{\boldsymbol{u}_\phi}\big)$, where it computes each of its components as follows. We denote

$$\mathsf{H}_{2\cdot3}(\mathsf{GID}, \boldsymbol{u}_\phi, j, k) = \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k)$$

for simplifying the representation of equations. For each $t \in S_\phi$ and $\mathcal{I}_{\boldsymbol{u}_\phi}$, it has four different cases to consider:

1. **Case 1** — $(t \in S_\phi \setminus \rho([\ell]))$ (i.e., the attribute is absent in the challenge policy (M, ρ)) — In this case, $\mathcal{B}$ simulates the secret keys according to the real experiment. It knows $\alpha_t, y_{t,j}$ for all $j \in \{2, \ldots, s_{\mathsf{max}}\}$ in clear and hence can compute

$$\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} = \Big(\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}\Big) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} \mathsf{H}_{2\cdot3}(\mathsf{GID}, \boldsymbol{u}_\phi, j, k)^{y_{t,j} u_{\iota_k}}$$

   where $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k)$ were sampled uniformly.
2. **Case 2** — $(t \in S_\phi \cap \rho([\ell]) \wedge \mathcal{I}_{\boldsymbol{u}_\phi} \neq \mathcal{I}^*)$ (i.e., the attribute is present in the challenge policy, but the associated index set does not match with the challenge index set) In this case, $\mathcal{B}$ extracts the corresponding exponents of the hash values from the list of hash queries and computes

$$\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} = \Big(\textstyle\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}\Big) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} \mathsf{H}_{2\cdot3}(\mathsf{GID}, \boldsymbol{u}_\phi, j, k)^{y_{t,j} u_{\iota_k}}$$

   where $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h''_{3,j,\iota_k}}$ were sampled uniformly from $\mathbb{Z}_q$.
3. **Case 3** — $(t \in S_\phi \cap \rho([\ell]) \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ and $\rho^{-1}(\mathcal{C} \cup S_\phi)$ constitutes an unauthorized subset of the rows of $\mathbf{M}$ (i.e., S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \rho)$). Note that the inner product value $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u}_\phi$ can be either zero or non-zero in this case. Since S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \rho)$, there exists a vector $\boldsymbol{d}_\phi = (d_{\phi,1}, \ldots, d_{\phi,s_{\mathsf{max}}}) \in \mathbb{Z}_q^{s_{\mathsf{max}}}$ such that $d_{\phi,1} = 1$ and the inner product $\boldsymbol{M}'_i \star \boldsymbol{d}_\phi = 0$ for all $i \in \rho^{-1}(S_\phi)$, where $\boldsymbol{M}'_i$ denotes the i-th row of $\mathbf{M}'$.

$\mathcal{B}$ computes the secret key $\mathsf{SK}_{t,\boldsymbol{u}}$ as follows.

$$\begin{aligned}
\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} &= (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} \mathsf{H}_{2.3}(\mathsf{GID}, \boldsymbol{u}_\phi, j, k)^{y_{t,j} u_{\iota_k}} \\
&= (\prod_{k=1}^{n} (g_2^{ab})^{\eta_k M'_{\rho^{-1}(t),1} u_{\iota_k}}) \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} (g_2^{ab})^{\eta_k d_{\phi,j} M'_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} (g_2^{ab})^{\eta_k d_{\phi,j} M'_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= \prod_{k=1}^{n} (g_2^{ab})^{\eta_k u_{\iota_k} (\boldsymbol{M}'_{\rho^{-1}(t)} \star \boldsymbol{d}_\phi)} \cdot g_2^{L_\phi(a,b)} = g_2^{L_\phi(a,b)}
\end{aligned}$$

where $L_\phi(a,b)$ represents a linear function in a, b and hence $g_2^{L_\phi(a,b)}$ can be efficiently computable by $\mathcal{B}$. The first equality follows from the definition of $\alpha_t, y_{t,j}$ (Equation (5.1)) and the hash functions H_1 (Equation (5.2)), H_2 (Equation (5.3) and Equation (5.4)) and H_3 (Equation (5.5) and Equation (5.6)). The last equality holds since $\boldsymbol{M}'_{\rho^{-1}(t)} \star \boldsymbol{d}_\phi = 0$ and the second last equality holds since $d_{\phi,1} = 1$.

4. **Case 4** — $(t \in S_\phi \cap \rho([\ell]) \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ and $\rho^{-1}(S_\phi)$ constitutes an authorized subset of rows of $\mathbf{M}$ (i.e., S_ϕ satisfies the challenge policy $(\mathbf{M}, \rho)$) – In this case, $\mathcal{B}$ computes the secret key $\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi}$ as follows.

$$\begin{aligned}
\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} &= (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} \mathsf{H}_{2.3}(\mathsf{GID}, \boldsymbol{u}_\phi, j, k)^{y_{t,j} u_{\iota_k}} \\
&= (\prod_{k=1}^{n} (g_2^{ab})^{\eta_k M'_{\rho^{-1}(t),1} u_{\iota_k}}) \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} ((g_2^{ab})^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}})^{M'_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= \left[(g_2^{ab})^{\eta_k M'_{\rho^{-1}(t),1}} \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} (g_2^{ab})^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j} M'_{\rho^{-1}(t),j}} \right]^{\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi} \cdot g_2^{L_\phi(a,b)} = g_2^{L_\phi(a,b)}
\end{aligned}$$

where the last equality follows from the fact that $\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi = 0$ if the secret key query satisfies the condition $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u}_\phi = 0$ as S_ϕ is authorized. Hence, in this case, $\mathcal{B}$ can efficiently simulates the secret key as $L_\phi(a,b)$ is linear in a, b.

Generating the Challenge Ciphertext: $\mathcal{B}$ implicitly sets the vectors

$$\begin{aligned}
\boldsymbol{z} &= -abc \cdot \boldsymbol{\eta} = -abc(\eta_1, \ldots, \eta_n) \in \mathbb{Z}_q^n, \\
\boldsymbol{x}_j &= -(ac, \ldots, ac) \in \mathbb{Z}_q^n, f_j = -ac \in \mathbb{Z}_q, \quad \forall j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}, \\
\boldsymbol{x}_j &= \mathbf{0} \in \mathbb{Z}_q^n, \quad f_j = 0 \in \mathbb{Z}_q, \quad \forall j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}
\end{aligned}$$

There are two cases to consider according to the authority whether it is corrupted or non-corrupted.

1. **Case 1** — $\rho(i) \in \mathcal{C}$ (meaning that the authority associated with this row is corrupted) — In this case, it holds that $\boldsymbol{M}'_i \mathbf{B} = \mathbf{0}$ and $M'_{i,j} \boldsymbol{x}_j = 0$ for all $(i,j) \in \rho^{-1}(\mathcal{C}) \times [\widehat{s_{\mathsf{max}}}]$ since $\boldsymbol{M}'_i|_{[\widehat{s_{\mathsf{max}}}]} = \{\overbrace{0, \ldots, 0}^{\widehat{s_{\mathsf{max}}}}\}$ and due to the above implicit setting of $\mathbf{B}, \boldsymbol{x}_j$. Thus, for each such row, $\mathcal{B}$ picks $r_i \leftarrow \mathbb{Z}_q$, and using the authority public key $\mathsf{PK}_{\rho(i)} = (\mathrm{Y}_{\rho(i),1}, \mathrm{Y}_{\rho(i),2}, \ldots, \mathrm{Y}_{\rho(i),s_{\mathsf{max}}})$ obtained

from $\mathcal{A}$, it computes

$$\begin{aligned} C_0 &= [\![\boldsymbol{v}_\beta + z]\!]_T, \;\; C_{1,i} = [\![\boldsymbol{M}'_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T = [\![\boldsymbol{\vartheta}_i]\!]_T, \;\; C_{2,i} = [\![r_i]\!]_1, \\ C_{3,i,j,k} &= e([\![M'_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i[\![\mathrm{Y}_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\ &= e(r_i[\![\mathrm{Y}_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\ C_{4,i,j} &= [\![M'_{i,j}f_j + \mathrm{Y}_{\rho(i),j}r_i]\!]_1 = [\![\mathrm{Y}_{\rho(i),j}r_i]\!]_1 \end{aligned}$$

for all $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}$ and $k \in [n]$, where $\boldsymbol{\vartheta}_i = (\vartheta_{i,1}, \ldots, \vartheta_{i,m})$ and

$$\vartheta_{i,k} = e(r_i[\![\mathrm{Y}_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*)).$$

2. **Case 2** — $\rho(i) \in \mathcal{N}$ (meaning that the authority associated with this row is uncorrupted) — Firstly, $\mathcal{B}$ sets $C_0 = [\![\boldsymbol{v}_\beta + \boldsymbol{z}]\!]_T$ where β is the challenge bit for $\mathcal{A}$. It also implicitly sets $r_i = c$ and the matrix $\mathbf{B} = (\boldsymbol{z}, \mathbf{0}, \cdots, \mathbf{0})^\top \in \mathbb{Z}_q^{s_{\mathsf{max}} \times n}$. This implies $\boldsymbol{M}'_i\mathbf{B} = M'_{i,1}\boldsymbol{z} = -M'_{i,1} \cdot abc \cdot \boldsymbol{\eta}$ and the k-th element of the vector is $(\boldsymbol{M}'_i\mathbf{B})_k = -M'_{i,1}abc\eta_k$. Recall that, for each $i \in [\ell]$, we have $\alpha_{\rho(i)} = \alpha'_{\rho(i)} + a \cdot M'_{i,1}$ and $y_{\rho(i),j} = y'_{\rho(i),j} + aM'_{i,j}$. Now, $\mathcal{B}$ implicitly computes the vector $\boldsymbol{\vartheta}_i \coloneqq (\vartheta_{i,1}, \ldots, \vartheta_{i,m})$ as

$$\begin{aligned} \vartheta_{i,k} &= e(r_i[\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*)) \\ &= e([\![c\alpha'_{\rho(i)} + ac \cdot M'_{i,1}]\!]_1, [\![b\eta_k + \sum_{\widehat{k}=1}^{n-1} h_{1,\widehat{k}}\lambda_{k,\widehat{k}} + h_{1,\rho(i),\iota_k}]\!]_2) \\ &= [\![bc\alpha'_{\rho(i)}\eta_k + M'_{i,1}abc\eta_k + (c\alpha'_{\rho(i)} + ac \cdot M'_{i,1})\mathfrak{h}_{1,i,k}]\!]_T \end{aligned}$$

where $\mathfrak{h}_{1,i,k} = \sum_{\widehat{k}=1}^{n-1} h_{1,\widehat{k}}\lambda_{k,\widehat{k}} + h_{1,\rho(i),\iota_k}$. We write $\boldsymbol{\mathfrak{h}}_{1,i} = (h_{1,\rho(i),\iota_k})_{k=1}^n$. Thus, for each $i \in [\ell]$, $\mathcal{B}$ sets $C_{2,i} = [\![c]\!]_1$ and computes

$$\begin{aligned} C_{1,i} &= [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T = [\![bc\alpha'_{\rho(i)}\boldsymbol{\eta} + (c\alpha'_{\rho(i)} + ac \cdot M'_{i,1})\boldsymbol{\mathfrak{h}}_{1,i}]\!]_T \\ &= e(g_1^c, g_2^b)^{\alpha'_{\rho(i)}\boldsymbol{\eta}} \cdot e(g_1^c, g_2)^{\alpha'_{\rho(i)}\boldsymbol{\mathfrak{h}}_i} \cdot e(g_1^c, g_2^a)^{M'_{i,1}\boldsymbol{\mathfrak{h}}_{1,i}} \end{aligned}$$

Next, $\mathcal{B}$ computes $C_{3,i,j,k}$ as follows. Recall that $C_{3,i,j,k}$ is a product of two pairing operations. Note that, $M'_{i,j}x_{j,k} = 0$ if $j \in \{\widehat{s_{\mathsf{max}}}+1, \ldots, s_{\mathsf{max}}\}$. Thus, for $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$, the first pairing is computed as

$$\begin{aligned} & e([\![M'_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\ &= e([\![M'_{i,j}x_{j,k}]\!]_1, (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,\rho(i),\iota_k}}) \\ &= [\![M'_{i,j}x_{j,k}b\eta_k d_j^+ + M'_{i,j}x_{j,k}\mathfrak{h}_{2,i,k}]\!]_T \end{aligned}$$

where $d_j^+ = \sum_{\phi=1}^{Q} d_{\phi,j}$ and $\mathfrak{h}_{2,i,k} = \sum_{\widehat{k}=1}^{n-1} h_{2,\widehat{k}}\lambda_{k,\widehat{k}} + h_{2,\rho(i),\iota_k}$. If $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$, the second pairing is computed as

$$\begin{aligned} & e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\ &= e([\![cy'_{\rho(i),j} + acM'_{i,j}]\!]_1, (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,\rho(i),\iota_k}}) \\ &= [\![bc(y'_{\rho(i),j} + aM'_{i,j})\eta_k d_j^+ + c(y'_{\rho(i),j} + aM'_{i,j})\mathfrak{h}_{2,i,k}]\!]_T \end{aligned}$$

Finally, for each $i \in [\ell], j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}, k \in [n]$, the ciphertext component $C_{3,i,j,k}$ is obtained as

$$C_{3,i,j,k} = e([\![M'_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*))$$
$$= [\![bcy'_{\rho(i),j}\eta_k d_j^+ + cy'_{\rho(i),j}\mathfrak{h}_{2,i,k}]\!]_T$$
$$= e(g_1^c, g_2^b)^{y'_{\rho(i),j}\eta_k d_j^+} \cdot e(g_1^c, g_2)^{y'_{\rho(i),j}\mathfrak{h}_{2,i,k}}$$

which $\mathcal{B}$ can compute as a part of the challenge ciphertext. Now, if $j \in \{\widehat{s_{\mathsf{max}}}+1, \ldots, s_{\mathsf{max}}\}$, recall that $y_{\rho(i),j}$ are known is clear and hence $\mathcal{B}$ computes $C_{3,i,j,k}$ as

$$C_{3,i,j,k} = e([\![M'_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*))$$
$$= e(r_i[\![y_{\rho(i),j}]\!]_1, [\![h'_{2,j,\iota_k}]\!]_2) = e(g_1^c, g_2)^{y_{\rho(i),j}h'_{2,j,\iota_k}}$$

for all $i \in [\ell], k \in [n]$. The last remaining part $C_{4,i,j}$ is given by

$$C_{4,i,j} = [\![M'_{i,j}f_j + y_{\rho(i),j}r_i]\!]_1 = [\![-acM'_{i,j} + cy'_{\rho(i),j} + acM'_{i,j}]\!]_1 = (g_1^c)^{y'_{\rho(i),j}}$$

if $i \in [\ell], j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$. Note that, $M'_{i,j}f_j = 0$ and $y_{\rho(i),j}$ are known in clear for $j \in \{\widehat{s_{\mathsf{max}}}+1, \ldots, s_{\mathsf{max}}\}$. Hence, $\mathcal{B}$ computes $C_{4,i,j}$ as

$$C_{4,i,j} = [\![M'_{i,j}f_j + y_{\rho(i),j}r_i]\!]_1 = [\![cy_{\rho(i),j}]\!]_1 = (g_1^c)^{y_{\rho(i),j}}$$

for each $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}$. Observe that, the elements $\mathbf{B}, \boldsymbol{x}_j, f_j$ and r_i are not properly distributed. Thus, $\mathcal{B}$ re-randomizes the ciphertext components using the algorithm CTRand described below before it sends to $\mathcal{A}$.

Ciphertext Re-randomization Algorithm: The algorithm described below provides properly distributed ciphertexts even if the randomness used within the ciphertexts inputted into the algorithm are not uniform. The algorithm uses only publicly available information to perform the re-randomization and hence rectify the distribution of the challenge ciphertext in the reduction.

$\mathsf{CTRand}((\mathbf{M}, \boldsymbol{\rho}), \mathsf{CT}, \mathsf{PK})$: The algorithm takes input an LSSS access policy $(\mathbf{M}, \rho)$, where $\mathbf{M} = (M_{i,j})_{\ell \times s_{\mathsf{max}}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and $\rho : [\ell] \to \mathsf{U}_{\mathsf{att}}$, a ciphertext $\mathsf{CT} = ((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i \in [\ell], j \in \{2,\ldots,s_{\mathsf{max}}\}, k \in [m]}, \mathcal{I}_{\boldsymbol{v}})$, and the public key components PK such that $\rho([\ell]) \subseteq \mathsf{U}_{\mathsf{att}}$.

1. Sample
   (a) $r'_1, \ldots, r'_\ell \leftarrow \mathbb{Z}_q; \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{s_{\mathsf{max}}} \in \mathbb{Z}_q^n; f'_2, \ldots, f'_{s_{\mathsf{max}}} \in \mathbb{Z}_q$,
   (b) $\mathbf{B}' = (\boldsymbol{z}', \boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_{s_{\mathsf{max}}})^\top \in \mathbb{Z}_q^{s_{\mathsf{max}} \times n}$,
2. Compute $C'_0 = C_0 \cdot [\![\boldsymbol{z}']\!]_T$.
3. For all $i \in [\ell]$, $j \in \{2, \ldots, s_{\mathsf{max}}\}$ and $k \in [n]$, compute

$$C'_{1,i} = C_{1,i} \cdot [\![\boldsymbol{M}_i\mathbf{B}' + \boldsymbol{\vartheta}'_i]\!]_T, \quad C'_{2,i} = C_{2,i} \cdot [\![r'_i]\!]_1,$$
$$C'_{3,i,j,k} = C_{3,i,j,k} \cdot e([\![M_{i,j}x'_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r'_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*))$$
$$C'_{4,i,j} = C_{4,i,j} \cdot [\![M_{i,j}f'_j + y_{\rho(i),j}r'_i]\!]_1$$

   where $\boldsymbol{\vartheta}'_i = (\vartheta'_{i,1}, \ldots, \vartheta'_{i,n})$ and $\vartheta'_{i,k} = e(r'_i[\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*))$.
4. Output the ciphertext
   $\mathsf{CT} = \big((\mathbf{M}, \rho), C'_0, \{C'_{1,i}, C'_{2,i,}, C'_{3,i,j,k}, C'_{4,i,j}\}_{i \in [\ell], j \in \{2,\ldots,s_{\mathsf{max}}\}, k \in [m]}, \mathcal{I}_{\boldsymbol{v}}\big)$.

Guess: If $\mathcal{A}$ guesses the challenge bit $\beta \in \{0, 1\}$ correctly, $\mathcal{B}$ returns 1; Otherwise $\mathcal{B}$ outputs 0. Now, observe that $\boldsymbol{z} = -\tau \cdot \boldsymbol{\eta}$ where $[\![\tau]\!]_T$ is the DBDH challenge element. If $\tau = abc$, then all the secret keys and the challenge ciphertext are distributed properly, in particular, the challenge ciphertext is an encryption of the message vector $\boldsymbol{v}_\beta$ for $\beta \leftarrow \{0, 1\}$. Therefore, in this case, $\mathcal{A}$ outputs $\beta' = \beta$ with probability $1/2 + \epsilon(\lambda)$ where $\epsilon(\lambda)$ is the advantage of $\mathcal{A}$ in the static security game of the $\mathsf{MA\text{-}ABUIPFE}$ scheme. On the other hand, if τ is a random element of $\mathbb{Z}_q$ then the

ciphertext element C_0 is uniformly random in $\mathbb{G}_T$, and hence from $\mathcal{A}$'s point of view there is no information of the challenge bit β in the challenge ciphertext. So, the probability of $\mathcal{A}$ outputting $\beta' = \beta$ is exactly $1/2$. Hence, by the guarantee of DBDH assumption, $\mathcal{A}$ has a non-negligible advantage against the proposed MA-ABUIPFE scheme in the static security game. This completes the proof. □

6 The Proposed Large Universe MA-ABUIPFE from L-DBDH

In this section, we describe the construction of our LMA-ABUIPFE scheme. The construction is in prime-order groups and additionally uses hash functions that are modelled as random oracles in the security proof just like our small universe construction.

6.1 The Construction

GlobalSetup$(1^\lambda, s_{\max})$: The global setup algorithm takes input the security parameter λ and a vector length n both in unary, and the maximum width of an LSSS matrix supported by the scheme $s_{\max} = s_{\max}(\lambda)$. It generates $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, e)$ and specify hash functions $\mathsf{H}_1 : \mathsf{U}_{\mathsf{att}} \times \mathbb{Z} \times \mathbb{Z}^* \to \mathbb{G}_2$, $\mathsf{H}_2 : [s_{\max}] \times \mathbb{Z} \times \mathbb{Z}^* \to \mathbb{G}_2$, $\mathsf{H}_3 : \mathcal{GID} \times \mathbb{Z}^* \times [s_{\max}] \times \mathbb{Z} \to \mathbb{G}_2$ and $\mathsf{R} : \mathsf{U}_{\mathsf{att}} \times [s_{\max}] \times \mathbb{Z}^* \to \mathbb{G}_2$ mapping strings $(t, j) \in \mathsf{U}_{\mathsf{att}} \times [s_{\max}]$ to elements in $\mathbb{G}_2$. It outputs a global parameter $\mathsf{GP} = (\mathsf{G}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{R})$.

LocalSetup(GP, θ): The authority setup algorithm takes input the global parameter GP and an authority index $\theta \in \mathcal{AU}$. It samples $\alpha_\theta, y_{\theta,2}, \ldots, y_{\theta,s_{\max}} \leftarrow \mathbb{Z}_q$ and outputs $\mathsf{PK}_\theta = (\llbracket \alpha_\theta \rrbracket_1, \llbracket y_{\theta,2} \rrbracket_1, \ldots, \llbracket y_{\theta,s_{\max}} \rrbracket_1)$ and $\mathsf{MSK}_\theta = (\alpha_\theta, y_{\theta,2}, \ldots, y_{\theta,s_{\max}})$.

KeyGen$(\mathsf{GP}, \mathsf{GID}, \mathsf{MSK}_\theta, t, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$: The key generation algorithm takes input GP, the user's global identifier GID, the authority's secret key MSK_θ, an attribute t controlled by the authority and a vector $\boldsymbol{u} \in \mathbb{Z}_q^{|\mathcal{I}_{\boldsymbol{u}}|}$. It samples $\tau_j \leftarrow \mathbb{Z}_p$ for $j \in [s_{\max}]$ and proceeds as follows:

1. Parse $\mathcal{I}_{\boldsymbol{u}} = \{\iota_1, \ldots, \iota_n\}$ and $\boldsymbol{u} = (u_{\iota_1}, \ldots, u_{\iota_n})$.
2. Compute $\mathsf{K}_{t,\boldsymbol{u}} = (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}})^{\alpha_\theta u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}}) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{y_{\theta,j} u_{\iota_k}}$.
3. Compute $\mathsf{SK}_{t,\boldsymbol{u}} = \mathsf{K}_{t,\boldsymbol{u}} \cdot \prod_{j=1}^{s_{\max}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}})^{\tau_j}$ and $\mathsf{Z}_t^{(j)} = \llbracket \tau_j \rrbracket_1 \; \forall \; j \in [s_{\max}]$.

Output $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \mathsf{SK}_{t,\boldsymbol{u}}, \mathsf{Z}_t^{(j)}, \mathcal{I}_{\boldsymbol{u}})$.

Encrypt$(\mathsf{GP}, (\mathbf{M}, \boldsymbol{\delta}), \{\mathsf{PK}_\theta\}, \boldsymbol{v}, \mathcal{I}_{\boldsymbol{v}})$: The encryption algorithm takes input the global parameter GP, an LSSS access structure $(\mathbf{M}, \delta)$ where $\mathbf{M} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\max}}$ and $\delta : [\ell] \to \mathsf{U}_{\mathsf{att}}$, a set $\{\mathsf{PK}_\theta\}$ of public keys for all the relevant authorities, and a message vector $\boldsymbol{v} \in \mathbb{Z}_q^m$. The function δ maps the row indices of $\mathbf{M}$ to attributes. We define the function $\rho : [\ell] \to \mathcal{AU}$ as $\rho(\cdot) = \mathsf{T}(\delta(\cdot))$ which maps row indices of $\mathbf{M}$ to authorities. The algorithm proceeds as follows:

1. Parse $\mathcal{I}_{\boldsymbol{v}} = \{\iota_1, \ldots, \iota_m\}$ and $\boldsymbol{v} = (v_{\iota_1}, \ldots, v_{\iota_m})$.
2. Sample $\{r_i \leftarrow \mathbb{Z}_q\}_{i \in [\ell]}$ and $\boldsymbol{f} = (f_2, \ldots, f_{s_{\max}}) \leftarrow \mathbb{Z}_q^{s_{\max}-1}$.
3. Sample $\boldsymbol{z}, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{s_{\max}}, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{s_{\max}} \leftarrow \mathbb{Z}_q^m$.
4. Set the matrix $\mathbf{B} = \left[\boldsymbol{z}, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{s_{\max}}\right]^\top_{s_{\max} \times m}$.
5. Compute $\vartheta_{i,k} = e(r_i \llbracket \alpha_{\rho(i)} \rrbracket_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}}))$ and set $\boldsymbol{\vartheta}_i \coloneqq (\vartheta_{i,1}, \ldots, \vartheta_{i,m})$.
6. Compute the following terms:

$$C_0 = \llbracket \boldsymbol{v} + \boldsymbol{z} \rrbracket_T, \quad C_{1,i} = \llbracket \boldsymbol{M}_i \mathbf{B} + \boldsymbol{\vartheta}_i \rrbracket_T, \quad C_{2,i} = \llbracket r_i \rrbracket_1,$$
$$C_{3,i,j,k} = e(\llbracket M_{i,j} x_{j,k} \rrbracket_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})) \cdot e(r_i \llbracket y_{\rho(i),j} \rrbracket_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})),$$
$$C_{4,i,j} = \llbracket M_{i,j} f_j + y_{\rho(i),j} r_i \rrbracket_1,$$

for all $i \in [\ell], j \in \{2, \ldots, s_{\max}\}, k \in [m]$.

7. Compute $C_{5,i,j} = \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}_{\boldsymbol{v}})^{r_i}$ for all $i \in [\ell], j \in [s_{\mathsf{max}}]$.
8. Output the ciphertext

$$\mathsf{CT} = \left((\mathbf{M}, \delta), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}, C_{5,i,1}, C_{5,i,j}\}_{\substack{j \in \{2,\ldots,s_{\mathsf{max}}\},\\ i \in [\ell], k \in [m]}}, \mathcal{I}_{\boldsymbol{v}} \right).$$

Decrypt(GP, GID, CT, $\{\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}}\}$): It takes input the public key PK, a secret key $\mathsf{SK}_{S,\boldsymbol{u}}$ for an attribute set $S \subseteq \mathsf{U}_{\mathsf{att}}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a ciphertext CT for an access structure $(\mathbf{M}, \delta)$ with $\mathbf{M} \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and a map $\delta : [\ell] \to \mathsf{U}_{\mathsf{att}}$.

Parse $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \mathsf{SK}_{t,\boldsymbol{u}}, \mathsf{Z}_t^{(j)}, \mathcal{I}_{\boldsymbol{u}})$, where $i \in [\ell]$ and $\mathsf{CT} = ((\mathbf{M}, \delta), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}, C_{5,i,1}, C_{5,i,j}\}_{i \in [\ell], j \in \{2,\ldots,s_{\mathsf{max}}\}, k \in [m]}, \mathcal{I}_{\boldsymbol{v}})$. Denote a set $I = \{i | \delta(i) \in S\} \subseteq [\ell]$. If $(1, 0, \ldots, 0)$ is not in the span of $\mathbf{M}_I$ (i.e., $\mathbf{M}$ restricted to the set of rows from I) or $\mathcal{I}_{\boldsymbol{u}} \neq \mathcal{I}_{\boldsymbol{v}}$ decryption returns $\perp$. Else, when S satisfies $(\mathbf{M}, \rho)$, the algorithm finds $\{w_i \in \mathbb{Z}_q\}_{i \in I}$ such that $(1, 0, \ldots, 0) = \sum_{i \in I} w_i \mathbf{M}_i$. It first computes

$$[\![\Lambda_i]\!]_T = \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}})$$

and outputs $\log_{g_T}([\![\Gamma]\!]_T)$ where $[\![\Gamma]\!]_T = C_0 \bullet \boldsymbol{u} \cdot [\![\mu]\!]_T$ and

$$[\![\mu]\!]_T = \left(\prod_{i \in I} \left[\frac{C_{1,i} \bullet \boldsymbol{u} \cdot [\![\Lambda_i]\!]_T \cdot \prod_{j=1}^{s_{\mathsf{max}}} e(\mathsf{Z}_{\delta(i)}^{(j)}, C_{5,i,j})}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i} \right)^{-1}.$$

6.2 Correctness

Consider a secret key $\mathsf{SK}_{\mathsf{GID},S,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \{\mathsf{SK}_{t,\boldsymbol{u}}, \mathsf{Z}_t^{(j)}\}_{t \in S}, \mathcal{I}_{\boldsymbol{u}})$ consisting of a set of attributes satisfying the LSSS access structure $(\mathbf{M}, \delta)$ associated with a ciphertext $\mathsf{CT} = ((\mathbf{M}, \delta), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}, C_{5,i,1}, C_{5,i,j}\}_{\substack{j \in \{2,\ldots,s_{\mathsf{max}}\},\\ i \in [\ell], k \in [m]}}, \mathcal{I}_{\boldsymbol{v}})$ such that $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I}$. In particular, the vector $(1, 0, \ldots, 0) \in \mathsf{rowspan}(\mathbf{M}_I)$ corresponding to the set of indices $I = \{i \in I | \delta(i) = t \in S\}$.
For each $i \in I$, we have the following:

$$\begin{aligned} e(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}) = & \prod_{k=1}^{n} e(g_1, \mathsf{H}_1(\delta(i) \parallel \iota_k \parallel \mathcal{I}))^{r_i \alpha_{\rho(i)} u_{\iota_k}} \cdot \\ & \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (e(g_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I})) \cdot e(g_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)))^{r_i y_{\rho(i),j} u_{\iota_k}} \cdot \\ & \prod_{j=1}^{s_{\mathsf{max}}} e(g_1, \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}))^{\tau_j r_i} \end{aligned}$$

For $i \in I, j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$,

$$u_{\iota_k} C_{3,i,j,k} = e([\![M_{i,j} x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{u_{\iota_k}} \cdot e(g_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{r_i y_{\rho(i),j} u_{\iota_k}}$$

For $i \in I, j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$,

$$\begin{aligned} & e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}}) \\ & = e([\![M_{i,j} f_j]\!]_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{u_{\iota_k}} \cdot e(g_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{r_i y_{\rho(i),j} u_{\iota_k}} \end{aligned}$$

For each $i \in I$,

$$e(\mathsf{Z}^{(j)}_{\delta(i)}, C_{5,i,j}) = e(g_1, \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}))^{\tau_j r_i}.$$

Finally, for each $i \in I$, we have $C_{1,i} = [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T$ and so

$$\frac{C_{1,i} \bullet \boldsymbol{u} \cdot [\![\Lambda_i]\!]_T \cdot \prod_{j=1}^{s_{\max}} e(\mathsf{Z}^{(j)}_{\delta(i)}, C_{5,i,j})}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}$$

$$= [\![\boldsymbol{M}_i\mathbf{B} \bullet \boldsymbol{u}]\!]_T \prod_{k=1}^{n} e(g_1, \mathsf{H}_1(\delta(i) \parallel \iota_k \parallel \mathcal{I}))^{r_i \alpha_{\rho(i)} u_{\iota_k}} \cdot$$

$$\frac{\prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k)^{u_{\iota_k}})) \cdot \prod_{j=1}^{s_{\max}} e(\mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}), g_2)^{\tau_j r_i}}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}$$

$$= [\![\boldsymbol{M}_i\mathbf{B} \bullet \boldsymbol{u}]\!]_T \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{u_{\iota_k}} \cdot e([\![M_{i,j}f_j]\!]_1, \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u} \parallel j \parallel \iota_k))^{u_{\iota_k}}$$

Since $\mathsf{SK}_{S,\boldsymbol{u}}$ corresponds to a set of qualified authorities, $\exists\{w_i \in \mathbb{Z}_q\}_{i\in I}$ such that $\sum_{i\in I} w_i \boldsymbol{M}_i\mathbf{B} \bullet \boldsymbol{u} = (1, 0, \ldots, 0)\mathbf{B} \bullet \boldsymbol{u} = \boldsymbol{z} \bullet \boldsymbol{u}$ and $\sum_{i\in I} w_i M_{i,j} = 0, \forall j \in \{2, \ldots, s_{\max}\}$. Hence, we have

$$\prod_{i\in I} \left[\frac{C_{1,i} \bullet \boldsymbol{u} \cdot [\![\Lambda_i]\!]_T \cdot \prod_{j=1}^{s_{\max}} e(\mathsf{Z}^{(j)}_{\delta(i)}, C_{5,i,j})}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i}$$

$$= [\![\sum_{i\in I} w_i \boldsymbol{M}_i\mathbf{B} \bullet \boldsymbol{u}]\!]_T = [\![\boldsymbol{z} \bullet \boldsymbol{u}]\!]_T$$

Finally, the message is recovered as $\log_{g_T}([\![\Gamma]\!]_T)$ where

$$[\![\Gamma]\!]_T = (C_0 \bullet \boldsymbol{u})/[\![\boldsymbol{z} \bullet \boldsymbol{u}]\!]_T = [\![\boldsymbol{v} \bullet \boldsymbol{u} + \boldsymbol{z} \bullet \boldsymbol{u}]\!]_T/[\![\boldsymbol{z} \bullet \boldsymbol{u}]\!]_T = [\![\boldsymbol{v} \bullet \boldsymbol{u}]\!]_T$$

6.3 Security Analysis

Theorem 6.1 *If the L-DBDH assumption holds, then all* PPT *adversaries have a negligible advantage in breaking the static security of the proposed* LMA-ABUIPFE *scheme in the random oracle model.*

We prove this theorem by showing that if there is any PPT adversary $\mathcal{A}$ who breaks the static security of MA-ABUIPFE then there is a PPT adversary $\mathcal{B}$ who solves the L-DBDH problem with a non-negligible advantage.

Suppose $\mathcal{A}$ breaks the static security of the LMA-ABUIPFE scheme with a non-negligible advantage. We then build a PPT adversary $\mathcal{B}$ that solves the L-DBDH problem with a non-negligible advantage. Suppose, $\mathcal{B}$ gets an instance

$$\left(\mathsf{G}, \begin{pmatrix} [\![b]\!]_1, [\![c]\!]_1, \\ [\![a]\!]_2, [\![b]\!]_2 \end{pmatrix}, \left\{ \begin{matrix} [\![a\mu_i]\!]_1, [\![c/\mu_i]\!]_1, \\ [\![a\mu_i]\!]_2 \end{matrix} \right\}_{i\in[L]}, \left\{ \begin{matrix} [\![c\mu_\iota/\mu_i]\!]_1, [\![ac\mu_\iota/\mu_i]\!]_1, \\ [\![ac\mu_\iota/\mu_i]\!]_2 \end{matrix} \right\}_{\substack{i,\iota\in[L], \\ i\neq\iota}}, [\![\tau]\!]_T \right)$$

of the L-DBDH problem where $\ell \leq L$, $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$ is a group description, the elements $a, b, c, \mu_i \leftarrow \mathbb{Z}_q$ are random integers, and the element $\tau \in \mathbb{Z}_q$ is either abc or a random element of $\mathbb{Z}_q$. The algorithm $\mathcal{B}$ works as follows:

Generating the Global Public Parameters: $\mathcal{B}$ sets the global public parameter $\mathsf{GP} = \mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ and simulates $\mathcal{A}$ on behalf of the LMA-ABUIPFE challenger which takes input 1^λ and GP.

Attacker's Queries: Upon initialization, the adversary $\mathcal{A}$ sends the following to $\mathcal{B}$:

(a) A set $\mathcal{C} \subset \mathcal{AU}$ of corrupt authorities and their respective public keys
$$\{\mathsf{PK}_\theta = (\mathrm{Y}_{\theta,1}, \mathrm{Y}_{\theta,2}, \ldots, \mathrm{Y}_{\theta,s_{\mathsf{max}}})\}_{\theta \in \mathcal{C}},$$
where $\mathrm{Y}_{\theta,1}, \mathrm{Y}_{\theta,2}, \ldots, \mathrm{Y}_{\theta,s_{\mathsf{max}}} \in \mathbb{G}_1$ for all $\theta \in \mathcal{C}$.

(b) A set $\mathcal{N} \subset \mathcal{AU} \setminus \mathcal{C}$ of non-corrupt authorities for which $\mathcal{A}$ requests the public keys.

(c) A collection of hash queries $\mathcal{H}_1 = \{(t, \iota_k, \mathcal{I}) : t \in \mathsf{U}_{\mathsf{att}}, \iota_k \in \mathbb{Z}, \mathcal{I} \subset \mathbb{N}\}$, $\mathcal{H}_2 = \{(j, \iota_k, \mathcal{I}) : j \in \{2, \ldots, s_{\mathsf{max}}\}, \iota_k \in \mathbb{Z}, \mathcal{I} \subset \mathbb{N}\}$ and $\mathcal{H}_3 = \{(\mathsf{GID}, \boldsymbol{u}, j, \iota_k) : \mathsf{GID} \in \mathcal{GID}, \boldsymbol{u} \in \mathbb{Z}^*, j \in \{2, \ldots, s_{\mathsf{max}}\}, \iota_k \in \mathbb{Z}\}$.

(d) A set $\mathcal{R} = \{(t, j, \mathcal{I})\} \subset \mathsf{U}_{\mathsf{att}} \times [s_{\mathsf{max}}]$ for each of which $\mathcal{A}$ queries the set of R values $\{\mathsf{R}(t \parallel 1 \parallel \mathcal{I}), \ldots, \mathsf{R}(t \parallel s_{\mathsf{max}} \parallel \mathcal{I})\}$
The hash functions are modeled as a random oracle in this proof.

(e) A set $\mathcal{Q} = \{(\mathsf{GID}, S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})\}$ of secret key queries with $\mathsf{GID} \in \mathcal{GID}, S \subseteq \mathsf{U}_{\mathsf{att}}$ such that $\mathsf{T}(S) \cap \mathcal{C} = \emptyset$, $\boldsymbol{u} \in \mathbb{Z}^{|\mathcal{I}_{\boldsymbol{u}}|}$ and $\mathcal{I}_{\boldsymbol{u}} \subset \mathbb{Z}$ where GIDs are distinct in each of these tuples.

(f) Two message vectors $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathbb{Z}_q^n$ having the same index set $\mathcal{I}^*$, and a challenge LSSS access policy $(\mathbf{M}, \delta)$ with $\mathbf{M} = (M_{i,j})_{\ell \times s_{\mathsf{max}}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$, $\delta : [\ell] \to \mathsf{U}_{\mathsf{att}}$ and satisfying the constraint that for each $(\mathsf{GID}, S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}}) \in \mathcal{Q}$, either $S \cup \bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta) \subseteq [\ell]$ constitutes an unauthorized subset of rows of the access matrix $\mathbf{M}$ or the secret key vector $\boldsymbol{u}$ satisfies the relation $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u} = 0$ whenever $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}^*$. Note that the set $\bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta)$ contains the attributes belonging to the corrupt authorities.

Before answering $\mathcal{A}$'s queries, the adversary $\mathcal{B}$ substitute the secret sharing matrix $\mathbf{M}$ with the matrix $\mathbf{M}'$ from Lemma 3.1 computed using $I_{\mathcal{C}} = \{i \in [\ell] : \delta(i) \in \bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta)\}$ as the unauthorized subset of rows. Lemma 3.1 guarantees the fact that if $\mathcal{B}$ uses $\mathbf{M}'$ instead of $\mathbf{M}$ in the simulation, the view of $\mathcal{A}$ in the simulated game is information theoretically the same as if $\mathcal{B}$ would have used the original matrix $\mathbf{M}$. Furthermore, Lemma 3.1 implies that if we assume the subspace spanned by $\mathbf{M}_i|_{i \in I_{\mathcal{C}}}$ has dimension $\widetilde{c}$, then so is the dimension of the subspace spanned by $\mathbf{M}'_i|_{i \in I_{\mathcal{C}}}$ and $M'_{i,j} = 0$ for all $(i, j) \in I_{\mathcal{C}} \times [s_{\mathsf{max}} - \widetilde{c}]$. $\mathcal{B}$ now proceeds to answer the queries of $\mathcal{A}$. Denote $\widehat{s_{\mathsf{max}}} = s_{\mathsf{max}} - \widetilde{c}$, where $\widetilde{c}$ is the dimension of the sequence spanned by the rows of $\mathbf{M}_i|_{i \in I_{\mathcal{C}}}$, the latter being the rows of $\mathbf{M}$ controlled by corrupted authorities, $\mathcal{C}$.

Note that $\mathcal{I}^*$ can be any subset of $\mathbb{Z}$ and w.l.o.g one can consider $\mathcal{I}^* = [n]$[6] for some $n \in \mathbb{N}$. Inspired by the proof techniques of prior works [ABCP15, SP19], the reduction first compute a basis of $(\boldsymbol{v}_0 - \boldsymbol{v}_1)^\perp$ as $\{\widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_{n-1}\}$. Then the set $\widetilde{\mathcal{S}} = \{\boldsymbol{v}_0 - \boldsymbol{v}_1, \widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_{n-1}\}$ form a basis of $\mathbb{Z}_q^n$. For any vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, if we represent it as the linear combination of the vectors in $\widetilde{\mathcal{S}}$ as
$$\boldsymbol{u} = \zeta \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) + \sum_{k=1}^{n-1} \zeta_k \widetilde{\boldsymbol{b}}_k, \quad \text{for some } \zeta, \zeta_k \in \mathbb{Z}_q$$
then $\zeta = 0$ whenever it holds that $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u} = 0$. Let $\boldsymbol{e}_k$ be the k-th vector in the standard basis of $\mathbb{Z}_q^n$. We write $\boldsymbol{e}_i$ for each $i \in [n]$ as
$$\boldsymbol{e}_i = \eta_i \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) + \sum_{k=1}^{n-1} \lambda_{i,k} \widetilde{\boldsymbol{b}}_k \quad \text{for some } \eta, \lambda_{i,k} \in \mathbb{Z}_q.$$

[6] In particular, we consider a map $\gamma : \mathcal{I}^* \to [n]$ and use $\gamma(\iota_k) = k$ throughout the security analysis.

Generating Authority Public Keys: There are two cases to consider:

1. **Case 1** — $\theta \notin \rho([\ell])$ (the authority is not present within the challenge access structure) — In this case, $\mathcal{B}$ executes the Setup algorithm according to the real experiment. It samples $\alpha_\theta, y_{\theta,2}, \ldots, y_{\theta,s_{\max}} \leftarrow \mathbb{Z}_q$ by itself, and computes the public key component corresponding to attribute t as $([\![\alpha_\theta]\!]_1, [\![y_{\theta,2}]\!]_1, \ldots, [\![y_{\theta,s_{\max}}]\!]_1)$.
2. **Case 2** — $\theta \in \rho([\ell]) \setminus \mathcal{C}$ (the authority is non-corrupted and appears in the challenge access structure) — In this case, $\mathcal{B}$ samples $\alpha'_\theta, y'_{\theta,2}, \ldots, y'_{\theta,s_{\max}} \leftarrow \mathbb{Z}_q$ and implicitly sets $\alpha_\theta = \alpha'_\theta + \sum_{i\in X} a\mu_i \cdot M'_{i,1}$ and $y_{\theta,j} = y'_{\theta,j} + \sum_{i\in X} a\mu_i M'_{i,j}$ for $j \in \{2, \ldots, \widehat{s_{\max}}\}$ and $y_{\theta,j} = y'_{\theta,j}$ for $j \in \{\widehat{s_{\max}} + 1, \ldots, s_{\max}\}$(these are well-defined as ρ is injective), and sets the public key elements w.r.t. attribute θ as $([\![\alpha_\theta]\!]_1, [\![y_{\theta,2}]\!]_1, \ldots, [\![y_{\theta,s_{\max}}]\!]_1)$. where the elements $[\![\alpha_\theta]\!]_1$ and $[\![y_{\theta,j}]\!]_1$ for $j \in \{2, \ldots, \widehat{s_{\max}}\}$ are computed as follows:

$$[\![\alpha_\theta]\!]_1 = [\![\alpha'_\theta]\!]_1 \cdot \prod_{i\in X} M'^{}_{i,1}[\![a\mu_i]\!]_1, \;\; [\![y_{\theta,j}]\!]_1 = [\![y'_{\theta,j}]\!]_1 \cdot \prod_{i\in X} M'^{}_{i,j}[\![a\mu_i]\!]_1 \tag{6.1}$$

for all $j \in [2, \widehat{s_{\max}}]$. Note that, α_θ and $\{y_{\theta,j}\}_{j\in\{2,\ldots,s_{\max}\}}$ are distributed uniformly over $\mathbb{Z}_q$ and hence each of these elements of the public key is properly distributed.

Answering Hash Queries:

1. $\mathsf{H_1}$ **queries.** If $(\mathsf{T}(t) \in \rho([\ell]) \wedge \iota_k \in \mathcal{I}^* \wedge \mathcal{I} = \mathcal{I}^*)$, then sample uniformly random elements $h_{1,\widehat{k}}, h_{1,t,\iota_k}$ from $\mathbb{Z}_q$ and set

$$\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}) = (g_2^b)^{\eta_k} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{1,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{1,t,\iota_k}}. \tag{6.2}$$

   Otherwise, if $(\mathsf{T}(t) \notin \rho([\ell]) \vee \mathsf{T}(t) \in \mathcal{C} \vee \iota_k \notin \mathcal{I}^* \vee \mathcal{I} \neq \mathcal{I}^*)$, then output a random $\mathbb{G}_2$ element, i.e., sample uniformly random element h'_{1,t,ι_k} from $\mathbb{Z}_q$ and set $\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}) = g_2^{h'_{1,t,\iota_k}}$. The reduction stores the hash queries for future use.
2. $\mathsf{H_2}$ **queries.** If $(\iota_k \in \mathcal{I}^* \wedge \mathcal{I} = \mathcal{I}^*)$, then sample uniformly random elements $h_{2,\widehat{k}}, h_{2,j,\iota_k}$ for $j \in \{2, \ldots, \widehat{s_{\max}}\}$ (in Eq. 6.3) and elements h'_{2,j,ι_k} for $j \in \{\widehat{s_{\max}} + 1, \ldots, s_{\max}\}$ from $\mathbb{Z}_q$ (in Eq. 6.4) and set

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,j,\iota_k}} \tag{6.3}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = g_2^{h'_{2,j,\iota_k}} \tag{6.4}$$

   where Q denotes the total number of *non-accepting* key queries $\{(S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi})\}_{\phi\in[Q]}$ made by the adversary in the case where $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$ but the attributes in S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \delta)$. Note that, for such secret key queries, there exists a vector $\boldsymbol{d}_\phi = (d_{\phi,1}, \ldots, d_{\phi,s_{\max}}) \in \mathbb{Z}_q^{s_{\max}}$ such that $d_{\phi,1} = 1$ and the inner product $\boldsymbol{M}'_i \cdot \boldsymbol{d}_\phi = 0$ for all i such that $\delta(i) \in S_\phi \cup \bigcup_{\theta\in\mathcal{C}} \mathsf{T}^{-1}(\theta)$, where $\boldsymbol{M}'_i$ denotes the i-th row of $\mathbf{M}'$. Additionally, the set of rows $\mathcal{R}_\mathbf{M} = \{\boldsymbol{M}'_i \in \mathbb{Z}_q^{s_{\max}} : i \in \rho^{-1}(\mathcal{C})\}$ has dimension c and $M'_{i,j} = 0$ for all $(i,j) \in \rho^{-1}(\mathcal{C}) \times [\widehat{s_{\max}}]$. Therefore, $\mathcal{R}_\mathbf{M}$ spans the entire subspace $\mathbb{V} = \left\{(\overbrace{0, \ldots, 0}^{\widehat{s_{\max}}}, \boldsymbol{\nu}) : \boldsymbol{\nu} \in \mathbb{Z}_q^c\right\}$. Thus, it follows that $\boldsymbol{d}_\phi$ is orthogonal to any of the vectors

$$\left\{(\overbrace{0, \ldots, 0}^{\widehat{s_{\max}}}, \overbrace{0, \ldots, 0}^{j-1}, 1, \overbrace{0, \ldots, 0}^{c-j})\right\}_{j\in\{\widehat{s_{\max}}+1,\ldots,s_{\max}\}}.$$

In other words, $d_{\phi,j} = 0$ for all $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$. Combining the above two facts, we have $(\boldsymbol{M}'_i|_{[\widehat{s_{\mathsf{max}}}]}) \cdot (\boldsymbol{d}_\phi|_{[\widehat{s_{\mathsf{max}}}]}) = 0$ for all $i \in [\ell]$ such that $\delta(i) \in S_\phi$, where for a vector $\boldsymbol{x}$, $\boldsymbol{x}|_X$ denotes a vector formed by taking the entries of $\boldsymbol{x}$ having indices in the set $X \in \mathbb{N}$. For simplicity of notation, let us denote $\boldsymbol{M}'_i \star \boldsymbol{d}_\phi = (\boldsymbol{M}'_i|_{[\widehat{s_{\mathsf{max}}}]}) \cdot (\boldsymbol{d}_\phi|_{[\widehat{s_{\mathsf{max}}}]})$ for $i \in \{i : \delta(i) \in S_\phi\}$. Otherwise, if $(\iota_k \notin \mathcal{I}^* \vee \mathcal{I} \neq \mathcal{I}^*)$, then output a random $\mathbb{G}_2$ element, i.e., sample uniformly random element h''_{2,t,ι_k} from $\mathbb{Z}_q$ and set $\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = g_2^{h''_{2,t,\iota_k}}$. The reduction stores the hash queries for future use.

3. **H_3 queries.** If $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $S_\phi \cap \rho([\ell]) \neq \emptyset$ and $S_\phi \cup \bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta)$ corresponds to an *unauthorized* subset of the rows of $\mathbf{M}$ then sample $h_{3,j,\iota_k} \leftarrow \mathbb{Z}_q$ and set

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = (g_2^b)^{\eta_k \sum_{\phi' \in [Q] \setminus \{\phi\}} -d_{\phi',j}} \cdot g_2^{h_{3,j,\iota_k}}, \ \forall\, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\} \tag{6.5}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h_{3,j,\iota_k}}, \ \forall\, j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\} \tag{6.6}$$

for all $\iota_k \in \mathcal{I}_{\boldsymbol{u}_\phi}$ such that $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$ and $\boldsymbol{d}_\phi$ is as defined above.
If $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $S_\phi \cap \delta([\ell]) \neq \emptyset$ and $\mathcal{I}_{\boldsymbol{u}_\phi} \neq \mathcal{I}^*$ then sample h'_{3,j,ι_k} uniformly at random from $\mathbb{Z}_q$ and set $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h'_{3,j,\iota_k}}$.
If $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $S_\phi \cap \delta([\ell]) \neq \emptyset$ and $S_\phi \cup \bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta)$ corresponds to an *authorized* subset of the rows of $\mathbf{M}$ then sample $h''_{3,j,\iota_k} \leftarrow \mathbb{Z}_q$ and set $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h''_{3,j,\iota_k}}$.
If $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$ and $\delta(i) \notin S_\phi \ \forall\, i \in [\ell]$ and $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$ then $\mathcal{B}$ sample $h'''_{3,j,\iota_k} \leftarrow \mathbb{Z}_q$ and set

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = (g_2^b)^{\eta_k \sum_{\phi \in [Q]} -d_{\phi,j}} \cdot g_2^{h'''_{3,j,\iota_k}}, \ \forall\, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\} \tag{6.7}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h'''_{3,j,\iota_k}}, \ \forall\, j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\} \tag{6.8}$$

The reduction stores the hash queries for future use.
For all other cases, the reduction simple outputs a uniformly random element from $\mathbb{G}_2$ to answer the hash query $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k)$.

4. **R queries.** If $\mathsf{T}(t) \notin \rho([\ell])$ or $\mathsf{T}(t) \in \mathcal{C}_\theta$ or $\mathcal{I} \neq \mathcal{I}^*$ then sample $\xi_{t,j} \leftarrow \mathbb{Z}_q$ and set $\mathsf{R}(t \parallel j \parallel \mathcal{I}) = g_2^{\xi_{t,j}}$ and stores the value to possibly reuse in a key query.
If $\mathsf{T}(t) \in \rho([\ell])$ and $\mathcal{I} = \mathcal{I}^*$ then define a set $X'' = \{i : \rho(i) = \mathsf{T}(t)\} \setminus \{i : \delta(i) = t\} \subseteq [\ell]$ and sample $\xi'_{t,j} \leftarrow \mathbb{Z}_q$ for each $j \in [s_{\mathsf{max}}]$ and set

$$\mathsf{R}(t \parallel j \parallel \mathcal{I}) = g_2^{\xi'_{t,j}} g_2^{\sum_{i \in X''} a\mu_i M'_{i,j}} = g_2^{\xi'_{t,j}} \prod_{i \in X''} (g_2^{a\mu_i})^{M'_{i,j}}. \tag{6.9}$$

Generating Secret Keys: For any $(\mathsf{GID}, S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$, $\mathcal{B}$ returns a secret key $\mathsf{SK}_{\mathsf{GID},S_\phi,\boldsymbol{u}_\phi} = (\mathsf{GID}, \boldsymbol{u}_\phi,$
$\{\mathsf{SK}_{t,\boldsymbol{u}_\phi}\}_{t \in S_\phi}, \mathcal{I}_{\boldsymbol{u}_\phi})$, where it computes each of its components as follows. For each $t \in S_\phi$ and $\mathcal{I}_{\boldsymbol{u}_\phi}$, it has four different cases to consider:

1. **Case 1** — $\mathsf{T}(t) = \theta \notin \rho([\ell])$ (i.e., the authority of the attribute is not present in the challenge policy) — In this case, $\mathcal{B}$ simulates the secret keys according to the real experiment. It knows $\alpha_\theta, y_{\theta,j}$ for all $j \in \{2, \ldots, s_{\mathsf{max}}\}$ in clear and hence can compute

$$\mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} = (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_\theta u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k))^{y_{\theta,j} u_{\iota_k}},$$

$$\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} = \mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\tau_j} \ \text{ and } \ \mathsf{Z}_{\phi,t}^{(j)} = [\![\tau_j]\!]_1 \ \forall\, j \in [s_{\mathsf{max}}]$$

where $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k)$ and $\tau_j \leftarrow \mathbb{Z}_q$ for $j \in [s_{\mathsf{max}}]$ were sampled uniformly and $\{\mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi})\}_{j \in [s_{\mathsf{max}}]}$ are computed using the procedure described above.

2. **Case 2** — $(\mathsf{T}(t) = \theta \in \rho([\ell]) \vee S_\phi \cap \delta([\ell]) = \emptyset \wedge \mathcal{I}_{\boldsymbol{u}_\phi} \neq \mathcal{I}^*)$ (i.e., the authority of the attribute is present in the challenge policy, but the associated index set does not match with the challenge index set) In this case, $\mathcal{B}$ extracts the corresponding exponents of the hash values from the list of hash queries and computes

$$\mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} = (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_\theta u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k))^{y_{\theta,j} u_{\iota_k}},$$

$$\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} = \mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\tau_j} \text{ and } \mathsf{Z}_{\phi,t}^{(j)} = [\![\tau_j]\!]_1 \; \forall \, j \in [s_{\mathsf{max}}]$$

where $\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k)$ and $\tau_j \leftarrow \mathbb{Z}_q$ for $j \in [s_{\mathsf{max}}]$ were sampled uniformly.

3. **Case 3** — $(\mathsf{T}(t) = \theta \in \rho([\ell]) \wedge S_\phi \cap \delta([\ell]) = \emptyset \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ (i.e., the authority of the attribute is present in $(\mathbf{M}, \delta)$, but none of user's attributes is in it) — In this case, according to the hash oracle queries, we have

$$\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = (g_2^b)^{\eta_k} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{1,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{1,t,\iota_k}}; \;\; \text{(Eq. 6.2)}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,j,\iota_k}}, \;\; \forall \, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}; \;\; \text{(Eq. 6.3)}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = g_2^{h'_{2,j,\iota_k}}, \;\; \forall \, j \in \{\widehat{s_{\mathsf{max}}}, \ldots, s_{\mathsf{max}}\}; \;\; \text{(Eq. 6.4)}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} -d_{\phi,j}} \cdot g_2^{h'''_{3,j,\iota_k}}, \; \forall \, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\} \;\; \text{(Eq. 6.7)}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h'''_{3,j,\iota_k}}, \; \forall \, j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\} \;\; \text{(Eq. 6.8)}$$

$$\mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = g_2^{\xi'_{t,j}} \cdot g_2^{\sum_{i \in X} a \mu_i M'_{i,j}}, \;\; \forall j \in [s_{\mathsf{max}}]. \;\; \text{(Eq. 6.9)}$$

Note that, in this case, $X'' = X = \{i : \rho(i) = \theta\} \subseteq [\ell]$. It sets $\mathsf{Z}_{\phi,t}^{(j)} = [\![\tau_j]\!]_1$ where $\tau_1 = -b \sum_{k=1}^{n} \eta_k u_{\iota_k}$ and $\tau_j \leftarrow \mathbb{Z}_q$ for all $j \in \{2, \ldots, s_{\mathsf{max}}\}$. Now, $\mathcal{B}$ computes the other secret key component $\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi}$ as

$$\begin{aligned}
&\mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\tau_j} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{\alpha_\theta u_{\iota_k}} \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} ((g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} -d_{\phi,j}})^{y_{\theta,j} u_{\iota_k}} \cdot \\
&\qquad\qquad (g_2^{\sum_{i \in X} a \mu_i M'_{i,1}})^{\tau_1} \cdot g_2^{L'(a\mu_i, b)} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{(\alpha'_\theta + \sum_{i \in X} a \mu_i M'_{i,1}) u_{\iota_k}} \cdot \prod_{i \in X} g_2^{-ba\mu_i M'_{i,1} \sum_{k=1}^{n} \eta_k u_{\iota_k}} \cdot g_2^{L'(a\mu_i, b)} \\
&= \prod_{k=1}^{n} \prod_{i \in X} g_2^{ab\eta_k \mu_i M'_{i,1} u_{\iota_k}} \cdot \prod_{i \in X} \prod_{k=1}^{n} g_2^{-ba\mu_i M'_{i,1} \eta_k u_{\iota_k}} \cdot g_2^{L(a,b)} = g_2^{L(a\mu_i, b)}
\end{aligned}$$

where $L'(a\mu_i, b), L(a\mu_i, b)$ denote linear functions in $a\mu_i, b$ and hence can be efficiently computable by $\mathcal{B}$. Note that, τ_1 is not properly distributed. $\mathcal{B}$ returns the re-randomized key using the SKRand algorithm described at the end of the reduction.

4. **Case 4** — $(\mathsf{T}(t) = \theta \in \rho([\ell]) \wedge S_\phi \cap \delta([\ell]) \neq \emptyset \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ and $S_\phi \cup \bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta)$ is unauthorized (i.e., the authority takes part in the execution of the challenge policy and the attributes along with the corrupted attributes form an unauthorized key)— In this case, according to the hash oracle queries, we have

$$\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = (g_2^b)^{\eta_k} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{1,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{1,t,\iota_k}}; \quad \text{(Eq. 6.2)}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,j,\iota_k}}, \ \forall\, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}; \quad \text{(Eq. 6.3)}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = g_2^{h'_{2,j,\iota_k}}, \ \forall\, j \in \{\widehat{s_{\mathsf{max}}}, \ldots, s_{\mathsf{max}}\}; \quad \text{(Eq. 6.4)}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = (g_2^b)^{\eta_k \sum_{\phi' \in [Q] \setminus \{\phi\}} -d_{\phi',j}} \cdot g_2^{h_{3,j,\iota_k}}, \ \forall\, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\} \quad \text{(Eq. 6.5)}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h_{3,j,\iota_k}}, \ \forall\, j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\} \quad \text{(Eq. 6.6)}$$

$$\mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = g_2^{\xi'_{t,j}} \cdot g_2^{\sum_{i \in X} a\mu_i M'_{i,j}}, \ \forall j \in [s_{\mathsf{max}}]. \quad \text{(Eq. 6.9)}$$

It sets $\mathsf{Z}_{\phi,t}^{(j)} = [\![\tau_j]\!]_1$ where $\tau_j = -bd_{\phi,j} \sum_{k=1}^{n} \eta_k u_{\iota_k}$ for $j \in \{1, \ldots, \widehat{s_{\mathsf{max}}}\}$ and $\tau_j \leftarrow \mathbb{Z}_q$ for all $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$. Now, $\mathcal{B}$ computes the other secret key component $\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi}$ as

$$\begin{aligned}
&\mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\tau_j} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{\alpha_\theta u_{\iota_k}} \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} ((g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot (g_2^b)^{\eta_k \sum_{\phi \in [Q] \setminus \{\phi\}} -d_{\phi',j}})^{y_{\theta,j} u_{\iota_k}} \cdot \\
&\qquad \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} (g_2^{\sum_{i \in X''} a\mu_i M'_{i,j}})^{\tau_j} \cdot g_2^{L'(a\mu_i, b)} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{\alpha_\theta u_{\iota_k}} \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} ((g_2^b)^{\eta_k d_{\phi,j}})^{y_{\theta,j} u_{\iota_k}} \cdot \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} (g_2^{\sum_{i \in X''} a\mu_i M'_{i,j}})^{\tau_j} \cdot g_2^{L'(a\mu_i, b)} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{(\alpha'_\theta + \sum_{i \in X} a\mu_i M'_{i,1}) u_{\iota_k}} \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} (g_2^{b\eta_k d_{\phi,j}})^{(y'_{\theta,j} + \sum_{i \in X} a\mu_i M'_{i,j}) u_{\iota_k}} \cdot \\
&\qquad \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} \prod_{i \in X''} g_2^{-ba\mu_i (M'_{i,j} d_{\phi,j}) \sum_{k=1}^{n} \eta_k u_{\iota_k}} \cdot g_2^{L'(a\mu_i, b)} \\
&= \prod_{k=1}^{n} \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} \prod_{i \in X} g_2^{ab\eta_k \mu_i (M'_{i,j} d_{\phi,j}) u_{\iota_k}} \cdot \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} \prod_{i \in X''} \prod_{k=1}^{n} g_2^{-ba\mu_i (M'_{i,j} d_{\phi,j}) \eta_k u_{\iota_k}} \cdot g_2^{L(a\mu_i, b)} \\
&= \prod_{k=1}^{n} \prod_{i \in X \setminus X''} g_2^{ab\eta_k \mu_i (\boldsymbol{M}'_i \star \boldsymbol{d}_\phi) u_{\iota_k}} \cdot g_2^{L(a\mu_i, b)} = g_2^{L(a\mu_i, b)}
\end{aligned}$$

where $L'(a\mu_i, b), L(a\mu_i, b)$ denote linear functions in $a\mu_i, b$ and hence can be efficiently computable by $\mathcal{B}$. Observe that, $X \setminus X'' = \{i : \delta(i) = t\} \subseteq [\ell]$ contains rows that map to t in the challenge policy. If $t \notin \delta([\ell])$, then $X \setminus X''$ is empty and hence the first product in the last equation is eliminated. On the other hand, if $t \in \delta([\ell])$ then $(\boldsymbol{d}_\phi \star \boldsymbol{M}'_i) = 0$ and hence the product in the last equation is vanished. Therefore, $\mathcal{B}$ can efficiently simulate the secret key.

Note that, τ_1 is not properly distributed. $\mathcal{B}$ returns the re-randomized key using the SKRand algorithm described at the end of the reduction.

5. **Case 5** — $(\mathsf{T}(t) = \theta \in \rho([\ell]) \wedge S_\phi \cap \delta([\ell]) \neq \emptyset \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ and $S_k \cup \bigcup_{\theta \in \mathcal{C}} \mathsf{T}^{-1}(\theta)$ is *authorized* (i.e., the authority takes part in the execution of the challenge policy and the attributes along with the corrupted attributes form an authorized key)– In this case, the hash oracles are defined as follows:

$$\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = (g_2^b)^{\eta_k} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{1,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{1,t,\iota_k}}; \quad \text{(Eq. 6.2)}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,j,\iota_k}}, \ \ \forall\, j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}; \quad \text{(Eq. 6.3)}$$

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = g_2^{h'_{2,j,\iota_k}}, \ \ \forall\, j \in \{\widehat{s_{\mathsf{max}}}, \ldots, s_{\mathsf{max}}\}; \quad \text{(Eq. 6.4)}$$

$$\mathsf{H}_3(\mathsf{GID} \parallel \boldsymbol{u}_\phi \parallel j \parallel \iota_k) = g_2^{h''_{3,j,k}}, \ \forall\, j \in \{2, \ldots, s_{\mathsf{max}}\}$$

$$\mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) = g_2^{\xi'_{t,j}} \cdot g_2^{\sum_{i \in X} a\mu_i M'_{i,j}}, \ \ \forall j \in [s_{\mathsf{max}}]. \quad \text{(Eq. 6.9)}$$

It sets $\mathsf{Z}_{\phi,t}^{(j)} = [\![\tau_j]\!]_1$ where $\tau_j = -bd_{\phi,j} \sum_{k=1}^{n} \eta_k u_{\iota_k}$ for $j \in \{1, \ldots, \widehat{s_{\mathsf{max}}}\}$ and $\tau_j \leftarrow \mathbb{Z}_q$ for all $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$. Now, $\mathcal{B}$ computes the other secret key component $\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi}$ as

$$\begin{aligned}
&\mathsf{K}_{\phi,t,\boldsymbol{u}_\phi} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\tau_j} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{\alpha_\theta u_{\iota_k}} \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} ((g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}})^{y_{\theta,j} u_{\iota_k}} \cdot \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} (g_2^{\sum_{i \in X''} a\mu_i M'_{i,j}})^{\tau_j} \cdot g_2^{L'(a\mu_i, b)} \\
&= \prod_{k=1}^{n} (g_2^{b\eta_k})^{(\alpha'_\theta + \sum_{i \in X} a\mu_i M'_{i,1}) u_{\iota_k}} \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{k=1}^{n} (g_2^{b\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}})^{(y'_{\theta,j} + \sum_{i \in X} a\mu_i M'_{i,j}) u_{\iota_k}} \cdot \\
&\qquad\qquad \prod_{j=1}^{\widehat{s_{\mathsf{max}}}} \prod_{i \in X''} g_2^{-ba\mu_i (M'_{i,j} d_{\phi,j}) \sum_{k=1}^{n} \eta_k u_{\iota_k}} \cdot g_2^{L'(a\mu_i, b)} \\
&= \left(\prod_{i \in X \setminus X''} g_2^{ab\mu_i M'_{i,1}} \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{i \in X} g_2^{ab\eta_k \mu_i (M'_{i,j} \sum_{\phi=1}^{Q} d_{\phi,j})} \cdot \prod_{j=2}^{\widehat{s_{\mathsf{max}}}} \prod_{i \in X''} g_2^{-ba\mu_i (M'_{i,j} d_{\phi,j})} \right)^{\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi} \cdot g_2^{L(a\mu_i, b)} \\
&= g_2^{L(a\mu_i, b)}
\end{aligned}$$

where $L'_\phi(a\mu_i, b), L_\phi(a\mu_i, b)$ denote linear functions in $a\mu_i, b$ and hence can be efficiently computable by $\mathcal{B}$. Observe that, for any authorized key it holds that $\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi = 0$ and hence the first product in the last equation is vanished. Therefore, $\mathcal{B}$ can efficiently simulate the secret key.
Note that, τ_1 is not properly distributed. $\mathcal{B}$ returns the re-randomized key using the SKRand algorithm described at the end of the reduction.

Generating the Challenge Ciphertext: $\mathcal{B}$ implicitly sets the vectors

$$\begin{aligned}
\boldsymbol{z} &= -abc \cdot \boldsymbol{\eta} = -abc(\eta_1, \ldots, \eta_n) \in \mathbb{Z}_q^n, \\
\boldsymbol{x}_j &= -(ac, \ldots, ac) \in \mathbb{Z}_q^n, f_j = -ac \in \mathbb{Z}_q, \quad \forall j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}, \\
\boldsymbol{x}_j &= \mathbf{0} \in \mathbb{Z}_q^n, \ \ f_j = 0 \in \mathbb{Z}_q, \quad \forall j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}
\end{aligned}$$

There are two cases to consider according to the authority whether it is corrupted or non-corrupted.

1. **Case 1** — $\rho(i) \in \mathcal{C}$ (meaning that the authority associated with this row is corrupted) — In this case, it holds that $\boldsymbol{M}'_i\mathbf{B} = \mathbf{0}$ and $M'_{i,j}\boldsymbol{x}_j = 0$ for all $(i,j) \in \rho^{-1}(\mathcal{C}) \times [\widehat{s_{\mathsf{max}}}]$ since $\boldsymbol{M}'_i|_{[\widehat{s_{\mathsf{max}}}]} = \left\{\overbrace{0,\ldots,0}^{\widehat{s_{\mathsf{max}}}}\right\}$ and due to the above implicit setting of $\mathbf{B}, \boldsymbol{x}_j$. Thus, for each such row, $\mathcal{B}$ picks $r_i \leftarrow \mathbb{Z}_q$, and using the authority public key $\mathsf{PK}_{\rho(i)} = (\mathrm{Y}_{\rho(i),1}, \mathrm{Y}_{\rho(i),2}, \ldots, \mathrm{Y}_{\rho(i),s_{\mathsf{max}}})$ obtained from $\mathcal{A}$, it computes

$$\begin{aligned}
C_0 &= [\![\boldsymbol{v}_\beta + \boldsymbol{z}]\!]_T,\ \ C_{1,i} = [\![\boldsymbol{M}'_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T = [\![\boldsymbol{\vartheta}_i]\!]_T,\ \ C_{2,i} = [\![r_i]\!]_1,\\
C_{3,i,j,k} &= e([\![M'_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i[\![\mathrm{Y}_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*))\\
&= e(r_i[\![\mathrm{Y}_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*))\\
C_{4,i,j} &= [\![M'_{i,j}f_j + \mathrm{Y}_{\rho(i),j}r_i]\!]_1 = [\![\mathrm{Y}_{\rho(i),j}r_i]\!]_1,\\
C_{5,i,1} &= \mathsf{R}(\delta(i) \parallel 1 \parallel \mathcal{I}^*)^{r_i},\quad C_{5,i,j} = \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}^*)^{r_i}
\end{aligned}$$

for all $i \in [\ell], j \in \{2,\ldots,s_{\mathsf{max}}\}$ and $k \in [n]$, where $\boldsymbol{\vartheta}_i = (\vartheta_{i,1},\ldots,\vartheta_{i,m})$ and

$$\vartheta_{i,k} = e(r_i[\![\mathrm{Y}_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*)).$$

2. **Case 2** — $\rho(i) \in \mathcal{N}$ (meaning that the authority associated with this row is uncorrupted) — Firstly, $\mathcal{B}$ sets $C_0 = [\![\boldsymbol{v}_\beta + \boldsymbol{z}]\!]_T$ where β is the challenge bit for $\mathcal{A}$. It also implicitly sets $r_i = c/\mu_i$ and the matrix $\mathbf{B} = (\boldsymbol{z}, \mathbf{0}, \cdots, \mathbf{0})^\top \in \mathbb{Z}_q^{s_{\mathsf{max}} \times n}$. This implies $\boldsymbol{M}'_i\mathbf{B} = M'_{i,1}\boldsymbol{z} = -M'_{i,1} \cdot abc \cdot \boldsymbol{\eta}$ and the k-th element of the vector is $(\boldsymbol{M}'_i\mathbf{B})_k = -M'_{i,1}abc\eta_k$. Recall that, for each $i \in [\ell]$, we have $\alpha_{\rho(i)} = \alpha'_{\rho(i)} + a\sum_{\varsigma\in X}\mu_\varsigma \cdot M'_{\varsigma,1}$ and $y_{\rho(i),j} = y'_{\rho(i),j} + a\sum_{\varsigma\in X}\mu_\varsigma M'_{\varsigma,j}$. Now, $\mathcal{B}$ implicitly computes the vector $\boldsymbol{\vartheta}_i \coloneqq (\vartheta_{i,1},\ldots,\vartheta_{i,m})$ as

$$\begin{aligned}
\vartheta_{i,k} &= e(r_i[\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*))\\
&= e([\![c\alpha'_{\rho(i)}/\mu_i + a\sum_{\varsigma\in X}(c\mu_\varsigma/\mu_i) \cdot M'_{\varsigma,1}]\!]_1, [\![b\eta_k + \sum_{\widehat{k}=1}^{n-1} h_{1,\widehat{k}}\lambda_{k,\widehat{k}} + h_{1,\rho(i),\iota_k}]\!]_2)\\
&= [\![bc\alpha'_{\rho(i)}\eta_k/\mu_i + ab\sum_{\varsigma\in X}(c\mu_\varsigma/\mu_i) \cdot M'_{\varsigma,1}\eta_k + \left(c\alpha'_{\rho(i)}/\mu_i + a\sum_{\varsigma\in X}(c\mu_\varsigma/\mu_i) \cdot M'_{\varsigma,1}\right)\mathfrak{h}_{1,i,k}]\!]_T
\end{aligned}$$

where $\mathfrak{h}_{1,i,k} = \sum_{\widehat{k}=1}^{n-1} h_{1,\widehat{k}}\lambda_{k,\widehat{k}} + h_{1,\rho(i),\iota_k}$. We write $\boldsymbol{\mathfrak{h}}_{1,i} = (h_{1,\rho(i),\iota_k})_{k=1}^n$. Thus, for each $i \in [\ell]$, $\mathcal{B}$ sets $C_{2,i} = [\![c/\mu_i]\!]_1$ and computes

$$\begin{aligned}
C_{1,i} &= [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T\\
&= [\![bc\alpha'_{\rho(i)}\boldsymbol{\eta}/\mu_i + b\sum_{\varsigma\in X\setminus\{i\}}(ac\mu_\varsigma/\mu_i) \cdot M'_{\varsigma,1}\boldsymbol{\eta} + \left(c\alpha'_{\rho(i)}/\mu_i + a\sum_{\varsigma\in X}(c\mu_\varsigma/\mu_i) \cdot M'_{\varsigma,1}\right)\boldsymbol{\mathfrak{h}}_{1,i}]\!]_T\\
&= e(g_1^{c/\mu_i}, g_2^b)^{\alpha'_{\rho(i)}\boldsymbol{\eta}} \cdot \prod_{\varsigma\in X\setminus\{i\}} e(g_1^{ac\mu_\varsigma/\mu_i}, g_2^b)^{M'_{\varsigma,1}\boldsymbol{\eta}} \cdot e(g_1^{c/\mu_i}, g_2)^{\alpha'_{\rho(i)}\boldsymbol{\mathfrak{h}}_{1,i}} \cdot \prod_{\varsigma\in X} e(g_1^{c/\mu_i}, g_2^{a\mu_\varsigma})^{M'_{\varsigma,1}\boldsymbol{\mathfrak{h}}_{1,i}}
\end{aligned}$$

Next, $\mathcal{B}$ computes $C_{3,i,j,k}$ as follows. Recall that $C_{3,i,j,k}$ is a product of two pairing operations. Note that, $M'_{i,j}x_{j,k} = 0$ if $j \in \{\widehat{s_{\mathsf{max}}}+1,\ldots,s_{\mathsf{max}}\}$. Thus, for $j \in \{2,\ldots,\widehat{s_{\mathsf{max}}}\}$, the first pairing is computed as

$$\begin{aligned}
&e([\![M'_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*))\\
&= e([\![M'_{i,j}x_{j,k}]\!]_1, (g_2^b)^{\eta_k\sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,\rho(i),\iota_k}})
\end{aligned}$$

$$= [\![M'_{i,j} x_{j,k} b \eta_k d_j^+ + M'_{i,j} x_{j,k} \mathfrak{h}_{2,i,k}]\!]_T$$

where $d_j^+ = \sum_{\phi=1}^{Q} d_{\phi,j}$ and $\mathfrak{h}_{2,i,k} = \sum_{\widehat{k}=1}^{n-1} h_{2,\widehat{k}} \lambda_{k,\widehat{k}} + h_{2,\rho(i),\iota_k}$. If $j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$, the second pairing is computed as

$$\begin{aligned}
&e(r_i [\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= e([\![c y'_{\rho(i),j}/\mu_i + a \sum_{\varsigma \in X} (c\mu_\varsigma/\mu_i) M'_{\varsigma,j}]\!]_1, (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,\rho(i),\iota_k}}) \\
&= [\![\left(y'_{\rho(i),j} bc/\mu_i + b \sum_{\varsigma \in X} (ac\mu_\varsigma/\mu_i) M'_{\varsigma,j} \right) \eta_k d_j^+ + \left(c y'_{\rho(i),j}/\mu_i + a \sum_{\varsigma \in X} (c\mu_\varsigma/\mu_i) M'_{\varsigma,j} \right) \mathfrak{h}_{2,i,k}]\!]_T
\end{aligned}$$

Finally, for each $i \in [\ell], j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}, k \in [n]$, the ciphertext component $C_{3,i,j,k}$ is obtained as

$$\begin{aligned}
&e([\![M'_{i,j} x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i [\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= [\![y'_{\rho(i),j} \eta_k d_j^+ bc/\mu_i + b \sum_{\varsigma \in X \setminus \{i\}} (ac\mu_\varsigma/\mu_i) M'_{\varsigma,j} \eta_k d_j^+ + y'_{\rho(i),j} \mathfrak{h}_{2,i,k} c/\mu_i + a \sum_{\varsigma \in X \setminus \{i\}} (c\mu_\varsigma/\mu_i) M'_{\varsigma,j} \mathfrak{h}_{2,i,k}]\!]_T \\
&= e(g_1^{c/\mu_i}, g_2^b)^{y'_{\rho(i),j} \eta_k d_j^+} \cdot \prod_{\varsigma \in X \setminus \{i\}} e(g_1^{ac\mu_\varsigma/\mu_i}, g_2^b)^{M'_{\varsigma,j} \eta_k d_j^+} \cdot e(g_1^{c/\mu_i}, g_2)^{y'_{\rho(i),j} \mathfrak{h}_{2,i,k}} \prod_{\varsigma \in X \setminus \{i\}} e(g_1^{c\mu_\varsigma/\mu_i}, g_2^a)^{M'_{\varsigma,j} \mathfrak{h}_{2,i,k}}
\end{aligned}$$

which $\mathcal{B}$ can compute as a part of the challenge ciphertext. Now, if $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$, recall that $y_{\rho(i),j}$ are known is clear and hence $\mathcal{B}$ computes $C_{3,i,j,k}$ as

$$\begin{aligned}
C_{3,i,j,k} &= e([\![M'_{i,j} x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i [\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= e(r_i [\![y_{\rho(i),j}]\!]_1, [\![h'_{2,j,\iota_k}]\!]_2) \\
&= e(g_1^{c/\mu_i}, g_2)^{y_{\rho(i),j} h'_{2,j,\iota_k}}
\end{aligned}$$

for all $i \in [\ell], k \in [n]$.
Next, $\mathcal{B}$ computes

$$\begin{aligned}
C_{4,i,j} &= [\![M'_{i,j} f_j + y_{\rho(i),j} r_i]\!]_1 \\
&= [\![-acM'_{i,j} + y'_{\rho(i),j} c/\mu_i + a \sum_{\varsigma \in X} M'_{\varsigma,j} c\mu_\varsigma/\mu_i]\!]_1 \\
&= (g_1^{c/\mu_i})^{y'_{\rho(i),j}} \prod_{\varsigma \in X \setminus \{i\}} (g_1^{ac\mu_\varsigma/\mu_i})^{M'_{\varsigma,j}}
\end{aligned}$$

if $i \in [\ell], j \in \{2, \ldots, \widehat{s_{\mathsf{max}}}\}$. Note that, $M'_{i,j} f_j = 0$ and $y_{\rho(i),j}$ are known in clear for $j \in \{\widehat{s_{\mathsf{max}}} + 1, \ldots, s_{\mathsf{max}}\}$. Hence, $\mathcal{B}$ computes $C_{4,i,j}$ as

$$C_{4,i,j} = [\![M'_{i,j} f_j + y_{\rho(i),j} r_i]\!]_1 = [\![y_{\rho(i),j} c/\mu_i]\!]_1 = (g_1^{c/\mu_i})^{y_{\rho(i),j}}$$

for each $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}$. Lastly, $\mathcal{B}$ computes

$$\begin{aligned}
C_{5,i,j} &= \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}^*)^{r_i} = (g_2^{\xi'_{\delta(i),j}} \cdot g_2^{\sum_{\varsigma \in X''} a\mu_\varsigma M'_{\varsigma,j}})^{c/\mu_i} \\
&= (g_2^{c/\mu_i})^{\xi'_{\delta(i),j}} \cdot \prod_{\varsigma \in X''} (g_2^{ac\mu_\varsigma/\mu_i})^{M'_{i,j}}
\end{aligned}$$

for all $i \in [\ell], j \in [s_{\mathsf{max}}]$. Notice that, $i \notin X''$ and $\ell \leq L$, and hence the adversary $\mathcal{B}$ can compute the ciphertext component $C_{5,i,j}$. Observe that, the elements $\mathbf{B}, \boldsymbol{x}_j, f_j$ and r_i are not properly distributed. Thus, $\mathcal{B}$ re-randomizes the ciphertext components using the algorithm CTRand described below before it sends to $\mathcal{A}$.

Re-randomization Algorithms: The algorithms described below provides properly distributed secret keys and ciphertexts even if the randomness used within their original version (inputted through the algorithm) are not uniform. The algorithm uses only publicly available information to perform the re-randomization. This algorithm is used to rectify the distribution of the secret keys and ciphertexts in our reduction.

$\mathsf{SKRand}(\mathsf{GP}, t, \mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}})$: The algorithm takes input the global parameters GP, an attribute t and a secret key $\mathsf{SK}_{\mathsf{GID},t,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \mathsf{SK}_{t,\boldsymbol{u}}, \{\mathsf{K}_t^{(j)}\}_{j\in[s_{\mathsf{max}}]}, \mathcal{I}_{\boldsymbol{u}})$. The algorithm samples $\tau'_j \leftarrow \mathbb{Z}_q$ for each $j \in [s_{\mathsf{max}}]$, computes

$$\mathsf{SK}'_{t,\boldsymbol{u}} = \mathsf{SK}_{t,\boldsymbol{u}} \cdot \prod_{j=1}^{s_{\mathsf{max}}} \mathsf{R}(t \parallel j \parallel \mathcal{I}_{\boldsymbol{u}})^{\tau'_j} \text{ and } \mathsf{Z}_{\phi,t}^{(j)'} = \mathsf{Z}_{\phi,t}^{(j)} \cdot [\![\tau'_j]\!]_1 \; \forall \, j \in [s_{\mathsf{max}}]$$

and outputs $\mathsf{SK}'_{\mathsf{GID},t,\boldsymbol{u}} = (\mathsf{GID}, \boldsymbol{u}, \mathsf{SK}'_{t,\boldsymbol{u}}, \{\mathsf{K}_t^{(j)'}\}_{j\in[s_{\mathsf{max}}]}, \mathcal{I}_{\boldsymbol{u}})$.

$\mathsf{CTRand}((\mathrm{M}, \rho), \mathsf{CT}, \mathsf{PK})$: The algorithm takes input an LSSS access policy $(\mathbf{M}, \delta)$, where $\mathbf{M} = (M_{i,j})_{\ell\times s_{\mathsf{max}}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell\times s_{\mathsf{max}}}$ and $\delta : [\ell] \to \mathsf{U}_{\mathsf{att}}$, a ciphertext $\mathsf{CT} = ((\mathbf{M}, \delta), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}, C_{5,i,1}, C_{5,i,j}\}_{\substack{j\in\{2,\ldots,s_{\mathsf{max}}\},,\\ i\in[\ell],k\in[m]}}, \mathcal{I}_{\boldsymbol{v}})$, and the public key components PK such that $\delta([\ell]) \subseteq \mathsf{U}_{\mathsf{att}}$. The algorithm defines a map $\rho : [\ell] \to \mathcal{AU}$ as before and proceeds as follows:

1. Sample
   (a) $r'_1, \ldots, r'_\ell \leftarrow \mathbb{Z}_q$,
   (b) $\mathbf{B}' = (\boldsymbol{z}', \boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_{s_{\mathsf{max}}})^\top \in \mathbb{Z}_q^{s_{\mathsf{max}}\times n}$,
   (c) $\boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{s_{\mathsf{max}}} \in \mathbb{Z}_q^n$,
   (d) $f'_2, \ldots, f'_{s_{\mathsf{max}}} \in \mathbb{Z}_q$
2. Compute $C'_0 = C_0 \cdot [\![\boldsymbol{z}']\!]_T$.
3. For all $i \in [\ell]$, $j \in \{2, \ldots, s_{\mathsf{max}}\}$ and $k \in [n]$, compute

$$\begin{aligned}
C'_{1,i} &= C_{1,i} \cdot [\![\boldsymbol{M}_i\mathbf{B}' + \boldsymbol{\vartheta}'_i]\!]_T, \\
C'_{2,i} &= C_{2,i} \cdot [\![r'_i]\!], \\
C'_{3,i,j,k} &= C_{3,i,j,k} \cdot e([\![M_{i,j}x'_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})) \cdot e(r'_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})) \\
C'_{4,i,j} &= C_{4,i,j} \cdot [\![M_{i,j}f'_j + y_{\rho(i),j}r'_i]\!]_1 \\
C'_{5,i,1} &= C_{5,i,1} \cdot \mathsf{R}(\delta(i) \parallel 1 \parallel \mathcal{I}_{\boldsymbol{v}})^{r'_i} \\
C'_{5,i,j} &= C_{5,i,j} \cdot \mathsf{R}(\delta(i) \parallel j \parallel \mathcal{I}_{\boldsymbol{v}})^{r'_i}
\end{aligned}$$

   where $\boldsymbol{\vartheta}'_i = (\vartheta'_{i,1}, \ldots, \vartheta'_{i,n})$ and $\vartheta'_{i,k} = e(r'_i[\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*))$.
4. Output the ciphertext $\mathsf{CT} = \left((\mathbf{M}, \rho), C'_0, \{C'_{1,i}, C'_{2,i,}, C'_{3,i,j,k}, C'_{4,i,j}, C'_{5,i,1}, C'_{5,i,j}\}_{\substack{j\in\{2,\ldots,s_{\mathsf{max}}\},,\\ i\in[\ell],k\in[m]}}, \mathcal{I}_{\boldsymbol{v}}\right)$.

Guess: If $\mathcal{A}$ guesses the challenge bit $\beta \in \{0, 1\}$ correctly, $\mathcal{B}$ returns 1; Otherwise $\mathcal{B}$ outputs 0. Now, observe that $\boldsymbol{z} = -\tau \cdot \boldsymbol{\eta}$ where $[\![\tau]\!]_T$ is the L-DBDH challenge element. If $\tau = abc$, then all the secret keys and the challenge ciphertext are distributed properly, in particular, the challenge ciphertext is an encryption of the message vector $\boldsymbol{v}_\beta$ for $\beta \leftarrow \{0, 1\}$. Therefore, in this case, $\mathcal{A}$

outputs $\beta' = \beta$ with probability $1/2 + \epsilon(\lambda)$ where $\epsilon(\lambda)$ is the advantage of $\mathcal{A}$ in the static security game of the LMA-ABUIPFE scheme. On the other hand, if τ is a random element of $\mathbb{Z}_q$ then the ciphertext element C_0 is uniformly random in $\mathbb{G}_T$, and hence from $\mathcal{A}$'s point of view there is no information of the challenge bit β in the challenge ciphertext. So, the probability of $\mathcal{A}$ outputting $\beta' = \beta$ is exactly $1/2$. Hence, by the guarantee of L-DBDH assumption, $\mathcal{A}$ has a non-negligible advantage against the proposed LMA-ABUIPFE scheme in the static security game. This completes the proof.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010. 7

ABCP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015. 3, 7, 11, 22, 31, 46

ABCP16. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Better security for functional encryption for inner product evaluations, 2016. 3

ABG19. Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 552–582. Springer, 2019. 7

ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 128–157. Springer, 2019. 3

ABP+17. Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. CCS '17, page 2277–2293, New York, NY, USA, 2017. Association for Computing Machinery. 3

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*, pages 467–497. Springer, 2020. 3

AGT21a. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 208–238. Springer, 2021. 3

AGT21b. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 224–255. Springer, 2021. 3, 4, 5, 6, 7, 11

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 333–362. Springer, 2016. 3, 7

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 440–456. Springer, 2005. 5, 13, 50, 51

BF01. Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, pages 213–229. Springer, 2001. 5, 7

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 514–532. Springer, 2001. 15

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011. 3

BSW13. Karyn Benson, Hovav Shacham, and Brent Waters. The k-bdh assumption family: Bilinear map cryptography from progressively weaker assumptions. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers'*

Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings, volume 7779 of *Lecture Notes in Computer Science*, pages 310–325. Springer, 2013. 4, 6

CGKW18. Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 503–534. Springer, 2018. 6

CSG+18. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 703–732. Springer, 2018. 7

DDM16. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 164–195. Springer, 2016. 3

DKW21a. Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority ABE for dnfs from LWE. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 177–209. Springer, 2021. 3, 7, 15, 16, 18

DKW21b. Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority ABE from NC[1] from computational-BDH. Cryptology ePrint Archive, Paper 2021/1325, ePrint 2021. 3, 4, 5, 7, 8, 11, 16, 18

DKW22. Pratish Datta, Ilan Komargodski, and Brent Waters. Fully adaptive decentralized multi-authority abe. Cryptology ePrint Archive, Paper 2022/1311, 2022. 3, 7

Fre10. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 44–61. Springer, 2010. 4, 6

Gay20. Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 95–120. Springer, 2020. 3

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, 2006. 5, 7

Gui13. Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, volume 7954 of *Lecture Notes in Computer Science*, pages 357–372. Springer, 2013. 4, 6

JLMS19. Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 251–281. Springer, 2019. 3

JLS19. Aayush Jain, Huijia Lin, and Amit Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. Cryptology ePrint Archive, Paper 2019/1252, 2019. 3

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. 3

Lew12. Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 318–335. Springer, 2012. 4, 6

LKK+18. Joohee Lee, Dongwoo Kim, Duhyeong Kim, Yongsoo Song, Junbum Shin, and Jung Hee Cheon. Instant privacy-preserving biometric authentication for hamming distance. *Cryptology ePrint Archive*, 2018. 3

LW11. Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*, pages 568–588. Springer, 2011. 3, 16

NPP22. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access control. 13791:95–125, 2022. 7

O'N10. Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Paper 2010/556, ePrint 2010. 3

OT20. Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based encryption and signatures. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 103-A(1):41–73, 2020. 3

PD21. Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from lwe. In *Progress in Cryptology–LATINCRYPT 2021: 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6–8, 2021, Proceedings 7*, pages 127–148. Springer, 2021. 3

RW15. Yannis Rouselakis and Brent Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, volume 8975 of *Lecture Notes in Computer Science*, pages 315–332. Springer, 2015. 3, 5, 7, 11, 13, 14, 16, 17, 18, 19

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. 5

SP19. Edouard Dufour Sans and David Pointcheval. Unbounded inner-product functional encryption with succinct keys. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 426–441. Springer, 2019. 5, 7, 9, 11, 15, 22, 31, 46

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 457–473. Springer, 2005. 5, 7

Tom19. Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. 11923:459–488, 2019. 3

Wat09. Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 619–636. Springer, 2009. 11

Wat11. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2011. 5, 7

Wee22. Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 217–241. Springer, 2022. 5

WWW22. Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I*, volume 13747 of *Lecture Notes in Computer Science*, pages 651–679. Springer, 2022. 3, 5, 7

ZR18. Kai Zhou and Jian Ren. Passbio: Privacy-preserving user-centric biometric authentication. *IEEE Trans. Inf. Forensics Secur.*, 13(12):3050–3063, 2018. 3

APPENDIX

A The Proposed Small Universe ABUIPFE from DBDH

In this section, we describe the formal construction and security analysis of our small universe single authority ABUIPFE scheme. The construction is in prime-order groups and enjoys succinct secret keys.The construction also makes use of hash functions which are modeled as random oracles in the security analysis.

A.1 The Construction

Setup$(1^\lambda, s_{\mathsf{max}}, \mathsf{U_{att}})$: The setup algorithm takes input the security parameter λ, the maximum width of an LSSS matrix supported by the scheme $s_{\mathsf{max}} = s_{\mathsf{max}}(\lambda)$, the vector length n in unary and the description of the attribute universe $\mathsf{U_{att}}$. It first generates $\mathsf{G} = (q, \mathbb{G}, \mathbb{G}_T, g, e)$. Consider two hash functions $\mathsf{H}_1 : \mathsf{U_{att}} \times \mathbb{Z} \times \mathbb{Z}^* \to \mathbb{G}_2$ and $\mathsf{H}_2 : [s_{\mathsf{max}}] \times \mathbb{Z} \times \mathbb{Z}^* \to \mathbb{G}_2$. Then for each attribute $t \in \mathsf{U_{att}}$, it samples the scalars $\alpha_t, y_{t,2}, \ldots, y_{t,s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q$, and outputs

$$\mathsf{PK} = \left(\mathsf{G}, \{[\![\alpha_t]\!]_1, \{[\![y_{t,j}]\!]_1\}_{j\in\{2,\ldots,s_{\mathsf{max}}\}}\}_{t\in\mathsf{U_{att}}}\right), \quad \mathsf{MSK} = \{\mathsf{G}, \{\alpha_t, \{y_{t,j}\}_{j\in\{2,\ldots,s_{\mathsf{max}}\}}\}_{t\in\mathsf{U_{att}}}\}$$

KeyGen$(\mathsf{MSK}, S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})$: The key generation algorithm takes input master secret key MSK, a set of attributes $S \subseteq \mathsf{U_{att}}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ with an associated index set $\mathcal{I}_{\boldsymbol{u}} \subset \mathbb{N}$. For each $t \in S$, it does the following:

1. Parse $\mathcal{I}_{\boldsymbol{u}} = \{\iota_1, \ldots, \iota_n\}$ and $\boldsymbol{u} = (u_{\iota_1}, \ldots, u_{\iota_n})$.
2. For each $j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$, compute $\mathsf{K}_{j,k} = [\![\mathrm{K}_{j,k}]\!]_2$, where $\mathrm{K}_{j,k} \leftarrow \mathbb{Z}_q$.
3. Compute

$$\mathsf{SK}_{t,\boldsymbol{u}} = (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}}) \cdot \mathsf{K}_{j,k})^{y_{t,j} u_{\iota_k}}$$

.

Output $\mathsf{SK}_{S,\boldsymbol{u}} = \left(\boldsymbol{u}, \{\mathsf{SK}_{t,\boldsymbol{u}}\}_{t\in S}, \{\mathsf{K}_{j,k}\}_{j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[n]}, \mathcal{I}_{\boldsymbol{u}}\right)$ as the secret key.

Encrypt$(\mathsf{PK}, (\mathbf{M}, \boldsymbol{\rho}), \boldsymbol{v}, \mathcal{I}_{\boldsymbol{v}})$: The encryption algorithm takes input the public key PK, an LSSS access structure $(\mathbf{M}, \rho)$ where $\mathbf{M} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and $\rho : [\ell] \to \mathsf{U_{att}}$, and a message vector $\boldsymbol{v} \in \mathbb{Z}_q^m$. The function ρ maps the row indices of $\mathbf{M}$ to attributes. We assume that ρ is an injective function, that is, an attribute is associated with at most one row of $\mathbf{M}$. The algorithm proceeds as follows:

1. Parse $\mathcal{I}_{\boldsymbol{v}} = \{\iota_1, \ldots, \iota_m\}$ and $\boldsymbol{v} = (v_{\iota_1}, \ldots, v_{\iota_m})$.
2. Sample $\{r_i \leftarrow \mathbb{Z}_q\}_{i\in[\ell]}$ and $\boldsymbol{f} = (f_2, \ldots, f_{s_{\mathsf{max}}}) \leftarrow \mathbb{Z}_q^{s_{\mathsf{max}}-1}$.
3. Sample $\boldsymbol{z}, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{s_{\mathsf{max}}}, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{s_{\mathsf{max}}} \leftarrow \mathbb{Z}_q^m$.
4. Set the following matrices:

$$\mathbf{B} = \left[\boldsymbol{z}, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{s_{\mathsf{max}}}\right]_{s_{\mathsf{max}}\times m}^\top, \quad \mathbf{X} = \left[\boldsymbol{x}_2, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_{s_{\mathsf{max}}}\right]_{(s_{\mathsf{max}}-1)\times m}^\top$$

5. Compute $\vartheta_{i,k} = e(r_i [\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}}))$ and set $\boldsymbol{\vartheta}_i := (\vartheta_{i,1}, \ldots, \vartheta_{i,m})$.

6. Compute the following terms:

$$C_0 = [\![\boldsymbol{v} + \boldsymbol{z}]\!]_T, \quad C_{1,i} = [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T, \quad C_{2,i} = [\![r_i]\!]_1,$$
$$C_{3,i,j,k} = e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})) \cdot e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{v}})),$$
$$C_{4,i,j} = [\![M_{i,j}f_j + y_{\rho(i),j}r_i]\!]_1$$

for all $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [m]$.

7. Output the ciphertext, $\mathsf{CT} = \left((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i\in[\ell],j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[m]}, \mathcal{I}_{\boldsymbol{v}}\right)$.

Decrypt($\mathsf{PK}, \mathsf{SK}_{S,\boldsymbol{u}}, \mathsf{CT}$): The decryption algorithm takes input the public key PK, a secret key $\mathsf{SK}_{S,\boldsymbol{u}}$ for an attribute set $S \subseteq \mathsf{U}_{\mathsf{att}}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a ciphertext CT for an access structure $(\mathbf{M}, \rho)$ with $\mathbf{M} \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and an injective map $\rho : [\ell] \to \mathsf{U}_{\mathsf{att}}$.

Parse $\mathsf{SK}_{S,\boldsymbol{u}} = \left(\boldsymbol{u}, \{\mathsf{SK}_{\rho(i),\boldsymbol{u}}\}_{\rho(i)\in S}, \mathcal{I}_{\boldsymbol{u}}\right)$, where $i \in [\ell]$ and $\mathsf{CT} = ((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i\in[\ell],j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[m]}, \mathcal{I}_{\boldsymbol{v}})$. Denote $I = \{i | \rho(i) \in S\} \subseteq [\ell]$. If $(1, 0, \ldots, 0)$ is not in the span of $\mathbf{M}_I$ (i.e., $\mathbf{M}$ restricted to the set of rows from I) or $\mathcal{I}_{\boldsymbol{u}} \neq \mathcal{I}_{\boldsymbol{v}}$ decryption returns $\perp$. Else, when S satisfies $(\mathbf{M}, \rho)$, the algorithm finds $\{w_i \in \mathbb{Z}_q\}_{i\in I}$ such that $(1, 0, \ldots, 0) = \sum_{i\in I} w_i\mathbf{M}_i$. It then computes

$$[\![\Gamma]\!]_T = C_0 \bullet \boldsymbol{u} \cdot \left(\prod_{i\in I} \left[\frac{C_{1,i} \bullet \boldsymbol{u} \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{K}_{j,k}^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i} \right)^{-1}$$

and outputs $\log_{g_T}([\![\Gamma]\!]_T)$.

A.2 Correctness

Consider a secret key $\mathsf{SK}_{S,\boldsymbol{u}} = \left(\boldsymbol{u}, \{\mathsf{SK}_{t,\boldsymbol{u}}\}_{t\in S}, \{\mathsf{K}_j\}_{j\in\{2,\ldots,s_{\mathsf{max}}\}}, \mathcal{I}_{\boldsymbol{u}}\right)$ consisting of a set of attributes satisfying the LSSS access structure $(\mathbf{M}, \rho)$ associated with a ciphertext $\mathsf{CT} = ((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i,}, C_{3,i,j,k}, C_{4,i,j}\}_{i\in[\ell],j\in\{2,\ldots,s_{\mathsf{max}}\},k\in[m]}, \mathcal{I}_{\boldsymbol{v}})$ such that $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_{\boldsymbol{v}} = \mathcal{I}$. In particular, the vector $(1, 0, \ldots, 0) \in \mathsf{rowspan}(\mathbf{M}_I)$ corresponding to the set of indices $I = \{i \in I | \rho(i) = t \in S\}$.

For each $i \in I$, we have the following:

$$e(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}) = \prod_{k=1}^{n} e(g_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}))^{r_i\alpha_{\rho(i)}u_{\iota_k}} \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (e(g_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I})) \cdot e(g_1, \mathsf{K}_{j,k}))^{r_i y_{\rho(i),j} u_{\iota_k}}$$

For $i \in I, j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$,

$$u_{\iota_k}C_{3,i,j,k} = e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{u_{\iota_k}} \cdot e(g_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{r_i y_{\rho(i),j} u_{\iota_k}}$$

For $i \in I, j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$,

$$e(C_{4,i,j}, \mathsf{K}_{j,k}^{u_{\iota_k}}) = e([\![M_{i,j}f_j]\!]_1, \mathsf{K}_{j,k})^{u_{\iota_k}} \cdot e(g_1, \mathsf{K}_{j,k})^{r_i y_{\rho(i),j} u_{\iota_k}}$$

Finally, for each $i \in I$, we have $C_{1,i} = [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T$ and so

$$\frac{C_{1,i} \cdot \boldsymbol{u} \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{K}_{j,k}^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}$$

$$= [\![\boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u}]\!]_T \prod_{k=1}^{n} e(g_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}))^{r_i \alpha_{\rho(i)} u_{\iota_k}} \cdot \frac{\prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{K}_{j,k}^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)}$$

$$= [\![\boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u}]\!]_T \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} e([\![M_{i,j} x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}))^{u_{\iota_k}} \cdot e([\![M_{i,j} f_j]\!]_1, \mathsf{K}_{j,k})^{u_{\iota_k}}$$

Since $\mathsf{SK}_{S,\boldsymbol{u}}$ corresponds to a set of qualified authorities, $\exists \{w_i \in \mathbb{Z}_q\}_{i \in I}$ such that $\sum_{i \in I} w_i \boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u} = (1, 0, \ldots, 0)\mathbf{B} \cdot \boldsymbol{u} = \boldsymbol{z} \cdot \boldsymbol{u}$ and $\sum_{i \in I} w_i M_{i,j} = 0, \forall j \in \{2, \ldots, s_{\max}\}$. Hence, we have

$$\prod_{i \in I} \left[\frac{C_{1,i} \cdot \boldsymbol{u} \cdot \prod_{j=2}^{s_{\max}} \prod_{k=1}^{n} (u_{\iota_k} \cdot C_{3,i,j,k} \cdot e(C_{4,i,j}, \mathsf{K}_{j,k}^{u_{\iota_k}}))}{e\left(\mathsf{SK}_{\rho(i),\boldsymbol{u}}, C_{2,i}\right)} \right]^{w_i}$$

$$= [\![\sum_{i \in I} w_i \boldsymbol{M}_i\mathbf{B} \cdot \boldsymbol{u}]\!]_T = [\![\boldsymbol{z} \cdot \boldsymbol{u}]\!]_T$$

Finally, the message is recovered as $\log_{g_T}([\![\Gamma]\!]_T)$ where

$$[\![\Gamma]\!]_T = (C_0 \cdot \boldsymbol{u})/[\![\boldsymbol{z} \cdot \boldsymbol{u}]\!]_T = [\![\boldsymbol{v} \cdot \boldsymbol{u} + \boldsymbol{z} \cdot \boldsymbol{u}]\!]_T / [\![\boldsymbol{z} \cdot \boldsymbol{u}]\!]_T = [\![\boldsymbol{v} \cdot \boldsymbol{u}]\!]_T$$

A.3 Security Analysis

Theorem A.1 *If the* DBDH *assumption holds, then all* PPT *adversaries have a negligible advantage in breaking static security of the proposed small universe* ABUIPFE *scheme in the random oracle model.*

Proof. We prove this theorem by showing that if there is any PPT adversary $\mathcal{A}$ who breaks the static security of ABUIPFE then there is a PPT adversary $\mathcal{B}$ who solves the DBDH problem with a non-negligible advantage.

Suppose, $\mathcal{B}$ gets an instance $(\mathsf{G}, [\![a]\!]_1, [\![c]\!]_1, [\![a]\!]_2, [\![b]\!]_2, [\![\tau]\!]_T)$ of the DBDH problem where $\mathsf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T,$
$g, e) \leftarrow \mathcal{G}(1^\lambda)$ is a group description, the elements $a, b, c \leftarrow \mathbb{Z}_q$ are random integers, and the element $\tau \in \mathbb{Z}_q$ is either abc or a random element of $\mathbb{Z}_q$. The algorithm $\mathcal{B}$ works as follows:

On input λ, $\mathcal{A}$ outputs $s_{\max}, \mathsf{U}_{\mathsf{att}}$ and queries the following.

Attacker's Queries: Upon initialization, the adversary $\mathcal{A}$ sends the following to $\mathcal{B}$:

(a) A collection of hash queries $\mathcal{H}_1 = \{(t, \iota_k, \mathcal{I}) : t \in \mathsf{U}_{\mathsf{att}}, \iota_k \in \mathbb{Z}, \mathcal{I} \subset \mathbb{N}\}$ and $\mathcal{H}_2 = \{(j, \iota_k, \mathcal{I}) : j \in \{2, \ldots, s_{\max}\}, \iota_k \in \mathbb{Z}, \mathcal{I} \subset \mathbb{N}\}$.
(b) A set $\mathcal{Q} = \{(S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}})\}$ of secret key queries with $S \subseteq \mathsf{U}_{\mathsf{att}}$, $\boldsymbol{u} \in \mathbb{Z}^{|\mathcal{I}_{\boldsymbol{u}}|}$ and $\mathcal{I}_{\boldsymbol{u}} \subset \mathbb{Z}$.

(c) Two message vectors $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathbb{Z}_q^n$ having the same index set $\mathcal{I}^*$, and a challenge LSSS access policy $(\mathbf{M}, \rho)$ with $\mathbf{M} = (M_{i,j})_{\ell \times s_{\max}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\max}}$ and $\rho : [\ell] \to \mathsf{U}_{\mathsf{att}}$ injective and satisfying the constraint that for each $(S, \boldsymbol{u}, \mathcal{I}_{\boldsymbol{u}}) \in \mathcal{Q}_{\boldsymbol{u}}$, either $\rho^{-1}(S) \subseteq [\ell]$ constitutes an unauthorized subset of rows of the access matrix $\mathbf{M}$ or the secret key vector $\boldsymbol{u}$ satisfies the relation $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u} = 0$ whenever $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}^*$.

Note that $\mathcal{I}^*$ can be any subset of $\mathbb{Z}$ and w.l.o.g one can consider $\mathcal{I}^* = [n]$[7] for some $n \in \mathbb{N}$. Inspired by the proof techniques of prior works [ABCP15, SP19], the reduction first compute a basis of $(\boldsymbol{v}_0 - \boldsymbol{v}_1)^\perp$ as $\{\widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_{n-1}\}$. Then the set $\widetilde{\mathcal{S}} = \{\boldsymbol{v}_0 - \boldsymbol{v}_1, \widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_{n-1}\}$ form a basis of $\mathbb{Z}_q^n$. For any vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, if we represent it as the linear combination of the vectors in $\widetilde{\mathcal{S}}$ as

$$\boldsymbol{u} = \zeta \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) + \sum_{k=1}^{n-1} \zeta_k \widetilde{\boldsymbol{b}}_k, \quad \text{for some } \zeta, \zeta_k \in \mathbb{Z}_q$$

then $\zeta = 0$ whenever it holds that $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u} = 0$. Let $\boldsymbol{e}_k$ be the k-th vector in the standard basis of $\mathbb{Z}_q^n$. We write $\boldsymbol{e}_i$ for each $i \in [n]$ as

$$\boldsymbol{e}_i = \eta_i \cdot (\boldsymbol{v}_0 - \boldsymbol{v}_1) + \sum_{k=1}^{n-1} \lambda_{i,k} \widetilde{\boldsymbol{b}}_k \quad \text{for some } \eta, \lambda_{i,k} \in \mathbb{Z}_q.$$

Generating Public Key: There are two cases to consider:

1. **Case 1** — $t \in \mathsf{U}_{\mathsf{att}} \setminus \rho([\ell])$ (i.e., attribute t is absent in the challenge policy $(\mathbf{M}, \rho)$) — In this case, $\mathcal{B}$ executes the Setup algorithm according to the real experiment. It samples $\alpha_t, y_{t,2}, \ldots, y_{t,s_{\max}} \leftarrow \mathbb{Z}_q$ by itself, and computes the public key component corresponding to attribute t as $([\![\alpha_t]\!]_1, [\![y_{t,2}]\!]_1, \ldots, [\![y_{t,s_{\max}}]\!]_1)$.
2. **Case 2** — $t \in \mathsf{U}_{\mathsf{att}} \cap \rho([\ell])$ (i.e., attribute t appears in the challenge policy $(\mathbf{M}, \rho)$) — In this case, $\mathcal{B}$ samples $\alpha'_t, y'_{t,2}, \ldots, y'_{t,s_{\max}} \leftarrow \mathbb{Z}_q$ and implicitly sets $\alpha_t = \alpha'_t + a \cdot M_{\rho^{-1}(t),1}$ and $y_{t,j} = y'_{t,j} + aM_{\rho^{-1}(t),j}$ for $j \in \{2, \ldots, s_{\max}\}$ (these are well-defined as ρ is injective), and sets the public key elements w.r.t. attribute t as $([\![\alpha_t]\!]_1, [\![y_{t,2}]\!]_1, \ldots, [\![y_{t,s_{\max}}]\!]_1)$. where the elements $[\![\alpha_t]\!]_1$ and $[\![y_{t,j}]\!]_1$ for $j \in \{2, \ldots, s_{\max}\}$ are computed as follows:

$$[\![\alpha_t]\!]_1 = [\![\alpha'_t]\!]_1 \cdot M_{\rho^{-1}(t),1}[\![a]\!]_1, [\![y_{t,j}]\!]_1 = [\![y'_{t,j}]\!]_1 \cdot M_{\rho^{-1}(t),j}[\![a]\!]_1 \tag{A.1}$$

for all $j \in [2, s_{\max}]$. Note that, α_t and $\{y_{t,j}\}_{j \in \{2,\ldots,s_{\max}\}}$ are distributed uniformly over $\mathbb{Z}_q$ and hence each of these elements of the public key is properly distributed.

Finally, it sends the public key as $\mathsf{PK} = \big(\mathsf{G}, \{[\![\alpha_t]\!]_1, \{[\![y_{t,j}]\!]_1\}_{j \in \{2,\ldots,s_{\max}\}}\}_{t \in \mathsf{U}_{\mathsf{att}}}\big)$.

Answering Hash Queries:

1. **H_1 queries.** If $(\iota_k \in \mathcal{I}^* \wedge \mathcal{I} = \mathcal{I}^*)$, then sample uniformly random elements $h_{1,\widehat{k}}, h_{1,t,\iota_k}$ from $\mathbb{Z}_q$ and set

$$\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}) = (g_2^b)^{\eta_k} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{1,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{1,t,\iota_k}}. \tag{A.2}$$

Otherwise, if $(\iota_k \notin \mathcal{I}^* \vee \mathcal{I} \neq \mathcal{I}^*)$, then output a random $\mathbb{G}_2$ element, i.e., sample uniformly random element h'_{1,t,ι_k} from $\mathbb{Z}_q$ and set $\mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}) = g_2^{h'_{1,t,\iota_k}}$. The reduction stores the hash queries for future use.

[7] In particular, we consider a map $\gamma : \mathcal{I}^* \to [n]$ and use $\gamma(k) = \iota_k$ throughout the security analysis.

2. **H_2 queries.** If $(\iota_k \in \mathcal{I}^* \wedge \mathcal{I} = \mathcal{I}^*)$, then sample uniformly random elements $h_{2,\widehat{k}}, h_{2,t,\iota_k}$ from $\mathbb{Z}_q$ and set

$$\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}} \lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,t,\iota_k}} \tag{A.3}$$

where Q denotes the total number of *non-accepting* key queries $\{(S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi})\}_{\phi \in [Q]}$ made by the adversary in the case where $\mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*$ but the attributes in S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \rho)$. Note that, for such secret key queries, there exists a vector $\boldsymbol{d}_\phi = (d_{\phi,1}, \ldots, d_{\phi,s_{\mathsf{max}}}) \in \mathbb{Z}_q^{s_{\mathsf{max}}}$ such that $d_{\phi,1} = 1$ and the inner product $\boldsymbol{M}_i \cdot \boldsymbol{d}_\phi = 0$ for all $i \in \rho^{-1}(S)$, where $\boldsymbol{M}_i$ denotes the i-th row of $\mathbf{M}$.
Otherwise, if $(\iota_k \notin \mathcal{I}^* \vee \mathcal{I} \neq \mathcal{I}^*)$, then output a random $\mathbb{G}_2$ element, i.e., sample uniformly random element h'_{2,t,ι_k} from $\mathbb{Z}_q$ and set $\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}) = g_2^{h'_{2,t,\iota_k}}$. The reduction stores the hash queries for future use.

Generating Secret Keys: For any $(S_\phi, \boldsymbol{u}_\phi, \mathcal{I}_{\boldsymbol{u}_\phi}) \in \mathcal{Q}$, $\mathcal{B}$ returns a secret key $\mathsf{SK}_{S_\phi, \boldsymbol{u}_\phi} = (\boldsymbol{u}_\phi, \{\mathsf{SK}_{t,\boldsymbol{u}_\phi}\}_{t \in S_\phi},$
$\{\mathsf{K}_{\phi,j,k}\}_{j \in \{2,\ldots,s_{\mathsf{max}}\}}, \mathcal{I}_{\boldsymbol{u}_\phi})$, where it computes each of its components as follows. For each $t \in S_\phi$ and $\mathcal{I}_{\boldsymbol{u}_\phi}$, it has four different cases to consider:

1. **Case 1** — $(t \in S_\phi \setminus \rho([\ell]))$ (i.e., the attribute is absent in the challenge policy (M, ρ)) — In this case, $\mathcal{B}$ simulates the secret keys according to the real experiment. It knows $\alpha_t, y_{t,j}$ for all $j \in \{2, \ldots, s_{\mathsf{max}}\}$ in clear and hence can compute

$$\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} = (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{K}_{\phi,j,k})^{y_{t,j} u_{\iota_k}}$$

where $\{\mathsf{K}_{\phi,j,k} \leftarrow \mathbb{G}_2\}_{j \in \{2,\ldots,s_{\mathsf{max}}\}, k \in [n]}$ are sampled uniformly.
2. **Case 2** — $(t \in S_\phi \cap \rho([\ell]) \wedge \mathcal{I}_{\boldsymbol{u}_\phi} \neq \mathcal{I}^*)$ (i.e., the attribute is present in the challenge policy, but the associated index set does not match with the challenge index set) In this case, $\mathcal{B}$ extracts the corresponding exponents of the hash values from the list of hash queries and computes

$$\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} = (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{K}_{j,k})^{y_{t,j} u_{\iota_k}}$$

where $\{\mathsf{K}_{\phi,j,k} = g_2^{k_{\phi,j,k}}\}_{j \in \{2,\ldots,s_{\mathsf{max}}\}, k \in [n]}$ and $k_{\phi,j,k}$'s are sampled uniformly from $\mathbb{Z}_q$.
3. **Case 3** — $(t \in S_\phi \cap \rho([\ell]) \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ and $\rho^{-1}(S_\phi)$ constitutes an unauthorized subset of the rows of $\mathbf{M}$ (i.e., S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \rho)$).
Note that the inner product value $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u}_\phi$ can be either zero or non-zero in this case. Since S_ϕ does not satisfy the challenge policy $(\mathbf{M}, \rho)$, there exists a vector $\boldsymbol{d}_\phi = (d_{\phi,1}, \ldots, d_{\phi,s_{\mathsf{max}}}) \in \mathbb{Z}_q^{s_{\mathsf{max}}}$ such that $d_{\phi,1} = 1$ and the inner product $\boldsymbol{M}_i \cdot \boldsymbol{d}_\phi = 0$ for all $i \in \rho^{-1}(S_\phi)$, where $\boldsymbol{M}_i$ denotes the i-th row of $\mathbf{M}$. $\mathcal{B}$ computes the secret key $\mathsf{SK}_{t,\boldsymbol{u}}$ as follows.
It first samples $k'_{\phi,j,k} \leftarrow \mathbb{Z}_q$ and sets

$$\mathsf{K}_{\phi,j,k} = (g_2^b)^{\eta_k \sum_{\phi' \in [Q] \setminus \{\phi\}} - d_{\phi',j}} \cdot g_2^{k'_{\phi,j,k}}$$

for all $j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$. Note that the fact $\mathsf{K}_{\phi,j,k} \in \mathbb{G}_2$ is randomly distributed since $k'_{\phi,j,k}$ is chosen uniformly at random from $\mathbb{Z}_q$. Next, it computes

$$\begin{aligned}
\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} &= (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{K}_{j,k})^{y_{t,j} u_{\iota_k}} \\
&= (\prod_{k=1}^{n} (g_2^{ab})^{\eta_k M_{\rho^{-1}(t),1} u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} ((g_2^{ab})^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot (g_2^{ab})^{\eta_k \sum_{\phi' \in [Q] \setminus \{\phi\}} -d_{\phi',j}})^{M_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= (\prod_{k=1}^{n} (g_2^{ab})^{\eta_k M_{\rho^{-1}(t),1} u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (g_2^{ab})^{\eta_k d_{\phi,j} M_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= \prod_{j=1}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (g_2^{ab})^{\eta_k d_{\phi,j} M_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= \prod_{k=1}^{n} (g_2^{ab})^{\eta_k u_{\iota_k} (\boldsymbol{M}_{\rho^{-1}(t)} \cdot \boldsymbol{d}_\phi)} \cdot g_2^{L_\phi(a,b)} \\
&= g_2^{L_\phi(a,b)}
\end{aligned}$$

where $L_\phi(a,b)$ represents a linear function in a, b and hence $g_2^{L_\phi(a,b)}$ can be efficiently computable by $\mathcal{B}$. The first equality follows from the definition of $\alpha_t, y_{t,j}$ (Equation (A.1)) and the hash functions H_1 (Equation (A.2)) and H_2 (Equation (A.3)). The last equality holds due to the fact that $\boldsymbol{M}_{\rho^{-1}(t)} \cdot \boldsymbol{d}_\phi = 0$ and the second last equality holds since $d_{\phi,1} = 1$.

4. **Case 4** — $(t \in S_\phi \cap \rho([\ell]) \wedge \mathcal{I}_{\boldsymbol{u}_\phi} = \mathcal{I}^*)$ and $\rho^{-1}(S)$ constitutes an authorized subset of rows of $\mathbf{M}$ (i.e., S satisfies the challenge policy $(\mathbf{M}, \rho)$) – In this case, $\mathcal{B}$ samples $k''_{\phi,j,k} \leftarrow \mathbb{Z}_q$ and sets $\mathsf{K}_{\phi,j,k} = g^{k''_{\phi,j,k}}$ for all $j \in \{2, \ldots, s_{\mathsf{max}}\}$. It then computes the secret key $\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi}$ as follows:

$$\begin{aligned}
\mathsf{SK}_{\phi,t,\boldsymbol{u}_\phi} &= (\prod_{k=1}^{n} \mathsf{H}_1(t \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi})^{\alpha_t u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} (\mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}_{\boldsymbol{u}_\phi}) \cdot \mathsf{K}_{j,k})^{y_{t,j} u_{\iota_k}} \\
&= (\prod_{k=1}^{n} (g_2^{ab})^{\eta_k M_{\rho^{-1}(t),1} u_{\iota_k}}) \cdot \prod_{j=2}^{s_{\mathsf{max}}} \prod_{k=1}^{n} ((g_2^{ab})^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}})^{M_{\rho^{-1}(t),j} u_{\iota_k}} \cdot g_2^{L_\phi(a,b)} \\
&= \left[(g_2^{ab})^{\eta_k M_{\rho^{-1}(t),1}} \cdot \prod_{j=2}^{s_{\mathsf{max}}} (g_2^{ab})^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j} M_{\rho^{-1}(t),j}} \right]^{\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi} \cdot g_2^{L_\phi(a,b)} \\
&= g_2^{L_\phi(a,b)}
\end{aligned}$$

where the last equality follows from the fact that $\boldsymbol{\eta} \cdot \boldsymbol{u}_\phi = 0$ if the secret key query satisfies the condition $(\boldsymbol{v}_0 - \boldsymbol{v}_1) \cdot \boldsymbol{u}_\phi = 0$ as S_ϕ is authorized. Hence, in this case, $\mathcal{B}$ can efficiently simulates the secret key as $L_\phi(a,b)$ is linear in a, b.

Generating the Challenge Ciphertext: $\mathcal{B}$ implicitly sets the vectors

$$\begin{aligned}
\boldsymbol{z} &= -abc \cdot \boldsymbol{\eta} = -abc(\eta_1, \ldots, \eta_n) \in \mathbb{Z}_q^n, \\
\boldsymbol{x}_j &= -(ac, \ldots, ac) \in \mathbb{Z}_q^n, \quad \forall j \in \{2, \ldots, s_{\mathsf{max}}\}
\end{aligned}$$

Firstly, $\mathcal{B}$ sets $C_0 = [\![\boldsymbol{v}_\beta + \boldsymbol{z}]\!]_T$ where β is the challenge bit for $\mathcal{A}$. It also implicitly sets $r_i = c, f_j = -ac$ and the matrix $\mathbf{B} = (\boldsymbol{z}, \mathbf{0}, \cdots, \mathbf{0})^\top \in \mathbb{Z}_q^{s_{\mathsf{max}} \times n}$. This implies $\boldsymbol{M}_i \mathbf{B} = M_{i,1} \boldsymbol{z} = -M_{i,1} \cdot abc \cdot \boldsymbol{\eta}$ and the k-th element of the vector is $(\boldsymbol{M}_i \mathbf{B})_k = -M_{i,1} abc \eta_k$. Recall that, for

each $i \in [\ell]$, we have $\alpha_{\rho(i)} = \alpha'_{\rho(i)} + a \cdot M_{i,1}$ and $y_{\rho(i),j} = y'_{\rho(i),j} + aM_{i,j}$. Now, $\mathcal{B}$ implicitly computes the vector $\boldsymbol{\vartheta}_i \coloneqq (\vartheta_{i,1}, \ldots, \vartheta_{i,m})$ as

$$\begin{aligned}
\vartheta_{i,k} &= e(r_i[\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= e([\![c\alpha'_{\rho(i)} + ac \cdot M_{i,1}]\!]_1, [\![b\eta_k + \sum_{\widehat{k}=1}^{n-1} h_{1,\widehat{k}}\lambda_{k,\widehat{k}} + h_{1,\rho(i),\iota_k}]\!]_2) \\
&= [\![bc\alpha'_{\rho(i)}\eta_k + M_{i,1}abc\eta_k + (c\alpha'_{\rho(i)} + ac \cdot M_{i,1})\mathfrak{h}_{1,i,k}]\!]_T
\end{aligned}$$

where $\mathfrak{h}_{1,i,k} = \sum_{\widehat{k}=1}^{n-1} h_{1,\widehat{k}}\lambda_{k,\widehat{k}} + h_{1,\rho(i),\iota_k}$. We write $\boldsymbol{\mathfrak{h}}_{1,i} = (h_{1,\rho(i),\iota_k})_{k=1}^{n}$. Thus, for each $i \in [\ell]$, $\mathcal{B}$ sets $C_{2,i} = [\![c]\!]_1$ and computes

$$\begin{aligned}
C_{1,i} &= [\![\boldsymbol{M}_i\mathbf{B} + \boldsymbol{\vartheta}_i]\!]_T \\
&= [\![bc\alpha'_{\rho(i)}\boldsymbol{\eta} + (c\alpha'_{\rho(i)} + ac \cdot M_{i,1})\boldsymbol{\mathfrak{h}}_{1,i}]\!]_T \\
&= e(g_1^c, g_2^b)^{\alpha'_{\rho(i)}\boldsymbol{\eta}} \cdot e(g^c, g)^{\alpha'_{\rho(i)}\boldsymbol{\mathfrak{h}}_i} \cdot e(g_1^c, g_2^a)^{M_{i,1}\boldsymbol{\mathfrak{h}}_{1,i}}
\end{aligned}$$

Next, $\mathcal{B}$ computes $C_{3,i,j,k}$ as follows. Recall that $C_{3,i,j,k}$ is a product of two pairing operations. The first pairing is computed as

$$\begin{aligned}
&e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= e([\![M_{i,j}x_{j,k}]\!]_1, (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,\rho(i),\iota_k}}) \\
&= [\![M_{i,j}x_{j,k}b\eta_k d_j^+ + M_{i,j}x_{j,k}\mathfrak{h}_{2,i,k}]\!]_T
\end{aligned}$$

where $d_j^+ = \sum_{\phi=1}^{Q} d_{\phi,j}$ and $\mathfrak{h}_{2,i,k} = \sum_{\widehat{k}=1}^{n-1} h_{2,\widehat{k}}\lambda_{k,\widehat{k}} + h_{2,\rho(i),\iota_k}$. The second pairing is computed as

$$\begin{aligned}
&e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= e([\![cy'_{\rho(i),j} + acM'_{i,j}]\!]_1, (g_2^b)^{\eta_k \sum_{\phi=1}^{Q} d_{\phi,j}} \cdot \prod_{\widehat{k}=1}^{n-1} g_2^{h_{2,\widehat{k}}\lambda_{k,\widehat{k}}} \cdot g_2^{h_{2,\rho(i),\iota_k}}) \\
&= [\![bc(y'_{\rho(i),j} + aM_{i,j})\eta_k d_j^+ + c(y'_{\rho(i),j} + aM_{i,j})\mathfrak{h}_{2,i,k}]\!]_T
\end{aligned}$$

Finally, for each $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}, k \in [n]$, the ciphertext component $C_{3,i,j,k}$ is obtained as

$$\begin{aligned}
C_{3,i,j,k} &= e([\![M_{i,j}x_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r_i[\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
&= [\![bcy'_{\rho(i),j}\eta_k d_j^+ + cy'_{\rho(i),j}\mathfrak{h}_{2,i,k}]\!]_T \\
&= e(g_1^c, g_2^b)^{y'_{\rho(i),j}\eta_k d_j^+} \cdot e(g_1^c, g_2)^{y'_{\rho(i),j}\mathfrak{h}_{2,i,k}}
\end{aligned}$$

which $\mathcal{B}$ can compute as a part of the challenge ciphertext. The last remaining part $C_{4,i,j}$ is given by

$$C_{4,i,j} = [\![M_{i,j}f_j + y_{\rho(i),j}r_i]\!]_1$$

$$= [\![-acM_{i,j} + cy'_{\rho(i),j} + acM'_{i,j}]\!]_1 = (g_1^c)^{y'_{\rho(i),j}}$$

Therefore, for each $i \in [\ell], j \in \{2, \ldots, s_{\mathsf{max}}\}$, $\mathcal{B}$ can simulate $C_{4,i,j}$. Note that, the elements $\mathbf{B}, \boldsymbol{x}_j, f_j$ and r_i are not properly distributed. Thus, $\mathcal{B}$ re-randomizes the ciphertext components using the algorithm CTRand described below before it sends to $\mathcal{A}$.

Ciphertext Re-randomization Algorithm: The algorithm described below provides properly distributed ciphertexts even if the randomness used within the ciphertexts inputted into the algorithm are not uniform. The algorithm uses only publicly available information to perform the re-randomization and hence rectify the distribution of the challenge ciphertext in the reduction.

$\mathsf{CTRand}((\mathbf{M}, \boldsymbol{\rho}), \mathsf{CT}, \mathsf{PK})$: The algorithm takes input an LSSS access policy $(\mathbf{M}, \rho)$, where $\mathbf{M} = (M_{i,j})_{\ell \times s_{\mathsf{max}}} = (\boldsymbol{M}_1, \ldots, \boldsymbol{M}_\ell)^\top \in \mathbb{Z}_q^{\ell \times s_{\mathsf{max}}}$ and $\rho : [\ell] \to \mathsf{U}_{\mathsf{att}}$, a ciphertext $\mathsf{CT} = ((\mathbf{M}, \rho), C_0, \{C_{1,i}, C_{2,i}, C_{3,i,j,k}, C_{4,i,j}\}_{\substack{j \in \{2,\ldots,s_{\mathsf{max}}\}, \\ i \in [\ell], k \in [m]}}, \mathcal{I}_{\boldsymbol{v}})$, and the public key components PK such that $\rho([\ell]) \subseteq \mathsf{U}_{\mathsf{att}}$. The algorithm proceeds as follows:

1. Sample
   (a) $r'_1, \ldots, r'_\ell \leftarrow \mathbb{Z}_q$,
   (b) $\mathbf{B}' = (\boldsymbol{z}', \boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_{s_{\mathsf{max}}})^\top \in \mathbb{Z}_q^{s_{\mathsf{max}} \times n}$,
   (c) $\boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{s_{\mathsf{max}}} \in \mathbb{Z}_q^n$,
   (d) $f'_2, \ldots, f'_{s_{\mathsf{max}}} \in \mathbb{Z}_q$
2. Compute $C'_0 = C_0 \cdot [\![\boldsymbol{z}']\!]_T$.
3. For all $i \in [\ell]$, $j \in \{2, \ldots, s_{\mathsf{max}}\}$ and $k \in [n]$, compute

$$\begin{aligned}
C'_{1,i} &= C_{1,i} \cdot [\![\boldsymbol{M}_i \mathbf{B}' + \boldsymbol{\vartheta}'_i]\!]_T, \\
C'_{2,i} &= C_{2,i} \cdot [\![r'_i]\!], \\
C'_{3,i,j,k} &= C_{3,i,j,k} \cdot e([\![M_{i,j} x'_{j,k}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \cdot e(r'_i [\![y_{\rho(i),j}]\!]_1, \mathsf{H}_2(j \parallel \iota_k \parallel \mathcal{I}^*)) \\
C'_{4,i,j} &= C_{4,i,j} \cdot [\![M_{i,j} f'_j + y_{\rho(i),j} r'_i]\!]_1
\end{aligned}$$

   where $\boldsymbol{\vartheta}'_i = (\vartheta'_{i,1}, \ldots, \vartheta'_{i,n})$ and $\vartheta'_{i,k} = e(r'_i [\![\alpha_{\rho(i)}]\!]_1, \mathsf{H}_1(\rho(i) \parallel \iota_k \parallel \mathcal{I}^*))$.
4. Output the ciphertext $\mathsf{CT} = \left((\mathbf{M}, \rho), C'_0, \{C'_{1,i}, C'_{2,i}, C'_{3,i,j,k}, C'_{4,i,j}\}_{\substack{j \in \{2,\ldots,s_{\mathsf{max}}\}, \\ i \in [\ell], k \in [m]}}, \mathcal{I}_{\boldsymbol{v}} \right)$.

Guess: If $\mathcal{A}$ guesses the challenge bit $\beta \in \{0, 1\}$ correctly, $\mathcal{B}$ returns 1; Otherwise $\mathcal{B}$ outputs 0. Now, observe that $\boldsymbol{z} = -\tau \cdot \boldsymbol{\eta}$ where $[\![\tau]\!]_T$ is the DBDH challenge element. If $\tau = abc$, then all the secret keys and the challenge ciphertext are distributed properly, in particular, the challenge ciphertext is an encryption of the message vector $\boldsymbol{v}_\beta$ for $\beta \leftarrow \{0, 1\}$. Therefore, in this case, $\mathcal{A}$ outputs $\beta' = \beta$ with probability $1/2 + \epsilon(\lambda)$ where $\epsilon(\lambda)$ is the advantage of $\mathcal{A}$ in the static security game of the $\mathsf{ABUIPFE}$ scheme. On the other hand, if τ is a random element of $\mathbb{Z}_q$ then the ciphertext element C_0 is uniformly random in $\mathbb{G}_T$, and hence from $\mathcal{A}$'s point of view there is no information of the challenge bit β in the challenge ciphertext. So, the probability of $\mathcal{A}$ outputting $\beta' = \beta$ is exactly $1/2$. Hence, by the guarantee of DBDH assumption, $\mathcal{A}$ has a non-negligible advantage against the proposed $\mathsf{ABUIPFE}$ scheme in the static security game. This completes the proof.

B Generic Security of the L-DBDH Assumption

We prove in this section that the L-DBDH assumption is generically secure following the generic proof template of Boneh, Boyen and Goh [BBG05].

Theorem B.1 *The L-DBDH assumption holds in the generic bilinear group model.*

Proof. We first fix the polynomials P, Q, R, f according to the Definition A.4 of [BBG05] below:

$$P = \left(1, b, c, \{a\mu_i, c/\mu_i\}_{i\in[L]}, \{c\cdot\mu_\iota/\mu_i, ac\cdot\mu_\iota/\mu_i\}_{\iota,i,\in[L],i\neq\iota}\right) = (p_i)_i,$$
$$Q = (1, a, b, \{a\mu_i\}_{i\in[L]}, \{ac\cdot\mu_\iota/\mu_i\}_{\iota,i,\in[L],i\neq\iota}) = (q_j)_j, \quad R = (1), \quad f = abc$$

It is easy to see that there does not exist a set of constants $\{x_{i,j}, y_{i,j}, z\} \subset \mathbb{Z}_q$ such that

$$f = \sum_{i,j} x_{i,j} p_i q_j + \sum_{i,j} y_{i,j} q_i q_j + z.$$

Thus, f is independent of (P, Q, R) according to the Definition A.4 of [BBG05]. For any $L \in \mathbb{N}$, we see that P contains $s = 3+2L+2L\cdot(L-1) = 2L^2+3$ tuples. The maximum total degrees of P, Q are $d_P = 4, d_Q = 4$ and hence $d = \max(d_P+d_Q, 2d_Q, d_R, d_f) = 8$. Therefore, by [BBG05, Theorem A.5], we have that any generic algorithm breaking the L-DBDH assumption with advantage $1/2$ must take time at least $\Omega(\sqrt{q/d} - s)$, where $s = 2L^2 + 3$.