New Constructions and Proof Methods for Large Universe Attribute-Based Encryption

Yannis Rouselakis
University of Texas at Austin
jrous@cs.utexas.edu

Brent Waters
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

We propose two large universe Attribute-Based Encryption constructions. In a large universe ABE construction any string can be used as an attribute and attributes need not be enumerated at system setup. Our first construction establishes a novel large universe Ciphertext-Policy ABE scheme on prime order bilinear groups, while the second achieves a significant efficiency improvement over the large universe Key-Policy ABE systems of Lewko-Waters and Lewko. Both schemes are selectively secure in the standard model under two "q-type" assumptions similar to ones used in prior works. Our work brings back "program and cancel" techniques to this problem.

We provide implementations and benchmarks of our constructions in Charm; a programming environment for rapid prototyping of cryptographic primitives.

1 Introduction

Traditionally, public key encryption schemes provided any user with the ability to share data with another specific user in a private manner. However, in many applications we would like to have the additional capability to encrypt data for a set of users according to a specific policy on their credentials. For example, one might want to store data in a public server such that only parties with credentials of specific forms are able to decrypt. Instead of encrypting the data once for each party it would be beneficial to be able to encrypt only once for all desired parties. This encryption notion, called *Attribute-Based Encryption* (ABE), was introduced by Sahai and Waters [35]. In this setting, each user possesses a set of attributes/credentials and a secret key that corresponds to these credentials. The encrypting party can define any Boolean formula on the possible attributes and a user can decrypt if and only if his attribute set satisfies the Boolean formula.

Several Attribute-Based constructions have been presented since then (see related work below). A common classification property is whether a system is a "small universe" or "large universe" constructions. In "small universe" constructions the size of the attribute space is polynomially bounded in the security parameter and the attributes were fixed at setup. Moreover, the size of the public parameters grew linearly with the number of attributes. In "large universe" constructions, on the other hand, the size of the attribute universe can be exponentially large and is thus desirable to have

Achieving the large universe property can be challenging. Different works either imposed restrictions on the expressiveness of the policies or were proved secure in the random oracle model. For example, in [20] a bound n was fixed at setup on the number of attributes that could be used while encrypting a message. For constructions that had no bounds on the expressiveness of policies and constant sized public parameters, the random oracle security model was used.

The above restrictions place undesirable burdens on the deployment of ABE schemes. If the designer of the system desires the benefits of avoiding the random oracle heuristic, he has to pick a specific bound for the expressiveness of the system at the setup time; either the size of the attribute universe or the bound on the policies. If the bound is too small, the system might exhaust its functionality and will have to be completely rebuilt. For example, consider the design of a framework that allows Attribute-Based Encryption in a huge multinational company and suppose that, as this company expands, a large number of new attributes have to be added to the system. If this number exceeds the bound set during the initial deployment of the system, then the company would have to re-deploy the (expanded) system and possibly re-encrypt all its data spending a huge amount of expenses. On the other hand, if the bound chosen is too big, the increased size of the public parameters will impose an unnecessary efficiency burden on all operations.

The first large universe constructions in the standard model were presented in the recent work of Lewko and Waters [25]. They presented the first large universe [1] KP-ABE construction, secure in the standard model. The system was proved *selectively* secure under static assumptions. They utilized the *dual system* framework on composite order groups to prove security.

While this framework is highly useful for the proofs, the actual constructions require use of bilinear groups of large composite order. As a result, these schemes sustain a significant efficiency overhead in comparison to prime order ABE constructions. In a recent result [22] building on [30, 32, 23, 17], one can actually "emulate" the effects of the composite order groups by creating special subspaces of vectors, called *dual vector spaces*, and construct a large universe KP-ABE scheme on prime order groups. While this improves the efficiency of the original construction, there is still a significant performance penalty due to the required size of the vectors. To the best of our knowledge, no large universe CP-ABE construction secure in the standard model (on prime or composite order groups) and with constant-size public parameters has been published.

Goals and Contributions We present new constructions and proof techniques for Large Universe ABE in the standard model. Departing from recent trends our work constructions that are proved *selectivity* secure using what is know as partitioning style techniques.

We believe that this is an interesting avenue to explore for two reasons. First, by considering selective

[1] The authors of [25] refer to their construction as an "unbounded" scheme. We prefer the more specific terminology of small or large universe as there are multiple parameters which may be "bounded" in a construction. Using the specific terminology helps to avoid ambiguity.

model of security we are able to get more efficient and more practical constructions. While full security is the strongest notion of security, we believe selective is still a meaningful notion and can be a reasonable trade off for performance in some circumstances. In addition, new partitioning proofs can give different and new insights into the security of a construction or style of construction.

Second, Lewko and Waters [26] recently showed a surprising connection between Dual System Encryption and older selective proofs. Prior fully secure ABE systems [23] required an additional (relative to selective schemes) limit t on the number of times an attribute could be used in a formula. The public parameters and ciphertext size for KP-ABE (key size for CP-ABE) grew proportionally to the bound t. Lewko and Waters showed that through a new "delayed parameter" variant of Dual System Encryption this limit could be done away with. An integral part of their proof was that it leveraged older "program and cancel" style techniques. Given this recent work, a reasonable conclusion is that developing selectively secure proofs might typically become a first step to developing full security. (We note that the large universe construction of [25] was only proved selectively secure.)

We aim to get practical large universe ABE schemes by adapting and expanding the system from [25] into the prime order setting. In proving security we go back to more traditional "program and cancel" techniques instead of the dual system framework.

We present two practical large universe ABE constructions (one CP-ABE and one KP-ABE) in prime order bilinear groups both selectively secure in the standard model under two different q-type assumptions. Our three main objectives in this work were *large universe constructions*, *efficiency*, and *security in the standard model*. Both schemes support a "large universe" attribute space and their public parameters consist of a constant number of group elements. No bounds or other restrictions are imposed on the monotonic Boolean formulas or the attribute sets used by the algorithms of the schemes; thus eliminating the need for design decisions at setup. The efficiency objective refrained us from using composite order groups or dual pairing vector spaces, while to achieve security in the standard model we relied to non static (q-type) assumptions and selective notions. These assumptions are non static in the sense that a polynomial number of terms is given to the adversary and therefore they are intuitively stronger than the static ones. However, the polynomial number of terms gives the ability to the simulator of the proof to embed the additional entropy in the constant number of public parameters. We showcase different techniques for harnessing the power of these assumptions to achieve our large universe constructions. Finally, we demonstrate the efficiency of our constructions by implementing our schemes. We compare performance results to other ABE schemes in prime order groups.

Our Techniques The techniques used to achieve our goals and prove the security of our schemes fall in the category of *partitioning* methodologies. In this setting the simulator of the reduction sets up the public parameters of the systems in such a way that the set of the possible policies (for KP-ABE) or the powerset of the attribute universe (for CP-ABE) is partitioned in two disjoint sets. One for which he can create the secret keys and answer the attackers' queries, and one for which this is not possible, where the challenge query should belong. Since we are dealing with selective security notions, the simulator knows in advance the required challenge set and therefore the suitable partition. However due to the fact that we are aiming for large universe ABE, which implies constant size public parameters, the simulator has to embed a polynomial amount of "challenge information" in them. This is achieved by utilizing the non static power of our assumptions. Namely, the assumptions' "size" depends on the size of the declared challenge query. The additional terms available to the simulator allow him to create all the necessary terms for the reduction.

Both our schemes work in a "layered" fashion in order to encrypt information securely and being able to decrypt. In the KP-ABE construction, which is simpler and directly inspired by the composite order construction of [25], two "layers" are employed: the "secret sharing" layer and the "attribute layer". The first layer is responsible for the sharing of the master secret key during the key generation algorithm and the storing of the blinding factor randomness during the encryption algorithm. The "attribute layer" holds information about the attributes used in both key generation and encryption phases. A "binder term" is utilized to connect the two layers in a secure way. In the CP-ABE construction the situation is slightly more complicated due to the fact that the policies are applied on the ciphertext side. As a result, the "sharing" is applied to the blinding factor randomness and not on the master secret key. Therefore, an additional "binder term" in the public parameters is being used to allow correct decryption using the master secret key. As we will see, the assumptions and the corresponding reductions follow closely this "layer" intuition.

Finally, we mention that both constructions use the "individual randomness" technique from [25] in the "attribute layer" to achieve the large universe functionality. The component for each attribute is masked by a different randomness and as a result no restrictions are imposed on the policies or the attributes, since each component is individually randomized.

1.1 Related Work

Attribute-Based Encryption was introduced by Sahai and Waters [35]. The refinement of the two notions was given in [20] and many CP-ABE and KP-ABE selectively secure constructions followed [6, 15, 19, 33, 34, 43]. Most of them work for non monotonic access structures with the exception of the schemes by Ostrovsky, Sahai, and Waters [33], who showed how to realize negation by incorporating specific revocation schemes into the GPSW construction. Fully secure constructions in the standard model were first provided by Okamoto and Takashima [32] and Lewko, Okamoto, Sahai, Takashima, and Waters [23]. The first large universe KP-ABE construction in the standard model was given in [25] (composite order groups). Okamoto and Takashima initiated the dual pairing vector space framework in various works [30, 31, 32], which lead to the first large universe KP-ABE construction in prime order group groups by Lewko [22]. Parameterized (non static) assumptions were introduced in [7] and used in several subsequent works [18, 43]. The problem of an environment with multiple central authorities in ABE was considered in [13, 14, 24], while several authors have presented schemes that do not address the problem of collusion resistance [40, 28, 11, 2, 3, 4].

We note that several techniques in ABE schemes have roots in Identity-Based Encryption [36, 8, 16, 7, 42, 18, 9]. Finally, we mention here the related concept of Predicate Encryption introduced by Katz, Sahai, and Waters [21] and further refined in [38, 37, 31, 23, 32, 10].

1.2 Organization

In section 2 we introduce some notation, provide background information about linear secret sharing schemes, and introduce the complexity assumption for our CP-ABE scheme. Section 3 contains the algorithms and the selective security definition for CP-ABE schemes. Our CP-ABE construction and the security proof are in section 4. Finally, implementations and efficiency results are presented in section 5. The assumption for our KP-ABE construction is in appendix A. In appendices B and C we present the KP-ABE algorithms with the security definition, and our KP-ABE construction with the security proof, respectively. The generic group model proofs of our assumptions are shown in appendix D.

2 Preliminaries

2.1 Notation

For $n \in \mathbb{N}$, we define $[n] \overset{\text{def.}}{=} \{1, 2, \ldots, n\}$. Also, for $n_1, n_2, \ldots, n_k \in \mathbb{N}$: $[n_1, n_2, \ldots, n_k] \overset{\text{def.}}{=} [n_1] \times [n_2] \times \ldots \times [n_m]$. When $\mathcal{S}$ is a set, we denote by $s \overset{\$}{\leftarrow} \mathcal{S}$ the fact that the variable s is picked uniformly at random from $\mathcal{S}$. We write $s_1, s_2, \ldots, s_n \overset{\$}{\leftarrow} \mathcal{S}$ as shorthand for $s_1 \overset{\$}{\leftarrow} \mathcal{S}, s_2 \overset{\$}{\leftarrow} \mathcal{S}, \ldots, s_n \overset{\$}{\leftarrow} \mathcal{S}$. By $\mathsf{negl}(n)$ we denote a negligible function in n and by PPT probabilistic polynomial-time. $\mathcal{F}(\mathcal{S}_1 \to \mathcal{S}_2)$ is defined to be the set of functions from set $\mathcal{S}_1$ to $\mathcal{S}_2$.

The set of matrices of size $m \times n$ with elements in $\mathbb{Z}_p$ is denoted by $\mathbb{Z}_p^{m \times n}$. Special subsets are the set of row vectors of length n: $\mathbb{Z}_p^{1 \times n}$, and column vectors of length n: $\mathbb{Z}_p^{n \times 1}$. A row vector is written explicitly as $(v_1, v_2, \ldots, v_n)$, while a column vector as $(v_1, v_2, \ldots, x_n)^{\perp}$. When $\vec{v}$ is a vector (of any type), we will denote by v_i the i-th element. We denote by $\langle \vec{v}_1, \vec{v}_2 \rangle$ the inner product of vector $\vec{v}_1$ with $\vec{v}_2$, where each vector can either be a row or a column vector.

2.2 Access Structures and Linear Secret-Sharing Schemes

In this subsection, we present the formal definitions of access structures and linear secret-sharing schemes introduced in [5], adapted to match our setting. Our constructions use these tools to encode authorized sets of attributes, encrypt, and decode.

Definition 2.1 (Access Structures [5]). *Let $\mathcal{U}$ be the attribute universe. An* access structure *on $\mathcal{U}$ is a collection $\mathbb{A}$ of non-empty sets of attributes, i.e. $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\ \}$. The sets in $\mathbb{A}$ are called the* authorized sets *and the sets not in $\mathbb{A}$ are called the* unauthorized sets*.*

Additionally, an access structure is called monotone *if $\forall B, C \in \mathbb{A}$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.*

The idea in the CP-ABE setting is that a user of the system possess a set of attributes that can be either authorized and he can decrypt, or unauthorized and he can not get any information from the ciphertext. In the KP-ABE setting, a ciphertext is encrypted using a set of attributes and each user has one or more keys for specific access structures. The ciphertext is meant to be decrypted only by users' keys where the set of attributes is authorized for the keys' access structures.

In our constructions, we only consider monotone access structures, which means that as a user (CP-ABE setting) acquires more attributes, he will not lose his possible decryption privileges. General access structures in large universe ABE can be realized by splitting the attribute universe in half and treating the attributes of one half as the negated versions of the attributes in the other half [20].

Definition 2.2 (Linear Secret-Sharing Schemes (LSSS) [5]). *Let p be a prime and $\mathcal{U}$ the attribute universe. A secret-sharing scheme Π with domain of secrets $\mathbb{Z}_p$ realizing access structures on $\mathcal{U}$ is* linear over $\mathbb{Z}_p$ *if*

1. *The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.*
2. *For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a matrix $M \in \mathbb{Z}_p^{\ell \times n}$, called the share-generating matrix, and a function ρ, that labels the rows of M with attributes from $\mathcal{U}$, i.e. $\rho \in \mathcal{F}([\ell] \to \mathcal{U})$, which satisfy the following:*

   During the generation of the shares, we consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)^{\perp}$, where $r_2, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_p$. Then the vector of ℓ shares of the secret s according to Π is equal to $M\vec{v} \in \mathbb{Z}_p^{\ell \times 1}$. The share $(M\vec{v})_j$ where $j \in [\ell]$ "belongs" to attribute $\rho(j)$.

   We will be referring to the pair (M, ρ) as the policy of the access structure $\mathbb{A}$.

According to [5], each secret-sharing scheme (not only the linear ones) should satisfy the *reconstruction requirement* (each authorized set can reconstruct the secret) and the *security requirement* (any unauthorized set cannot reveal any partial information about the secret).

In our setting, let $\mathcal{S}$ denote an authorized set for the access structure $\mathbb{A}$ encoded by the policy (M, ρ). Then let I be the set of rows whose labels are in $\mathcal{S}$, i.e. $I = \{i | i \in [\ell] \wedge \rho(i) \in \mathcal{S}\}$. The reconstruction requirement asserts that the vector $(1, 0, \ldots, 0)$ is in the span of rows of M indexed by I. This means that there exist constants $\{\omega_i\}_{i \in I}$ in $\mathbb{Z}_p$ such that for any valid shares $\{\lambda_i = (M\vec{v})_i\}_{i \in I}$ of a secret s according to Π, it is true that: $\sum_{i \in I} \omega_i \lambda_i = s$. Additionally, it has been proved in [5] that the constants $\{\omega_i\}_{i \in I}$ can be found in time polynomial in the size of the share-generating matrix M.

On the other hand, for unauthorized sets $\mathcal{S}'$ no such constants $\{\omega_i\}$ exist. Moreover, in this case it is also true that if $I' = \{i | i \in [\ell] \wedge \rho(i) \in \mathcal{S}'\}$, there exists a vector $\vec{w} \in \mathbb{Z}_p^{1 \times n}$, such that its first component w_1 is any non zero element in $\mathbb{Z}_p$ and $\langle M_i, \vec{w} \rangle = 0$ for all $i \in I'$, where $M_i = (M_{i,1}, M_{i,2}, \ldots, M_{i,n})$; the i-th row of M.

Finally, we note that if the access structure is encoded as a monotonic Boolean formula over attributes[2], there is a generic algorithm that generates the corresponding access policy in polynomial time [5, 24].

[2]A monotonic Boolean formula consists of only AND and OR gates, for example $A_1 \wedge (A_2 \vee A_3)$. This means that as a key (in KP-ABE) or a ciphertext (in CP-ABE) acquires more attributes it will not lose the decryption capabilities.

2.3 Assumption 1

For our CP-ABE construction we will use a q-type assumption on prime order bilinear groups, denoted by q-1, which is similar to the Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption [43]. It is parameterized by a security parameter $\lambda \in \mathbb{N}$ and an integer q, polynomial in λ. We assume that there exists a group generator algorithm $\mathcal{G}(1^\lambda) \to (p, \mathbb{G}, \mathbb{G}_T, e)$ that outputs the description of the (symmetric) bilinear group of order $p = \Theta(2^\lambda)$. The generic security of the assumption is shown in appendix D. It is defined via the following game between a challenger and an attacker:

Initially the challenger calls the group generation algorithm with input the security parameter, picks a random group element $g \xleftarrow{\$} \mathbb{G}$, and $q+2$ random exponents $a, s, b_1, b_2, \ldots, b_q \xleftarrow{\$} \mathbb{Z}_p$. Then he sends to the attacker the group description $(p, \mathbb{G}, \mathbb{G}_T, e)$ and all of the following terms:

$$\begin{array}{ll}
g, g^s & \\
g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i/b_j^2} & \forall (i,j) \in [q,q] \\
g^{a^i/b_j} & \forall (i,j) \in [2q,q] \text{ with } i \neq q+1 \\
g^{a^i b_j/b_{j'}^2} & \forall (i,j,j') \in [2q,q,q] \text{ with } j \neq j' \\
g^{sa^i b_j/b_{j'}}, g^{sa^i b_j/b_{j'}^2} & \forall (i,j,j') \in [q,q,q] \text{ with } j \neq j'
\end{array}$$

In the challenge phase, the challenger flips a random coin $b \xleftarrow{\$} \{0,1\}$ and if $b = 0$, it gives to the attacker the term $e(g,g)^{sa^{q+1}}$. Otherwise it gives a random term $R \xleftarrow{\$} \mathbb{G}_T$. Finally the attacker outputs a guess $b' \in \{0,1\}$.

Definition 2.3. *We say that the q-1 assumption holds if all PPT attackers have at most a negligible advantage in λ in the above security game, where the advantage is defined as* $\mathsf{Adv} = \Pr[b' = b] - 1/2$.

Remark: Notice the absence of the term g^{a^{q+1}/b_j} in the third line of the assumption. If this term were given to the attacker, then he could break the assumption trivially by pairing it with the corresponding g^{sb_j} term. On the other hand, the term $g^{a^{q+1}b_j/b_{j'}^2}$ is given, and this poses no problems in the generic group model since $j \neq j'$ and by possible pairing the adversary cannot get rid of the b_j's. See appendix D for further details.

3 Ciphertext-Policy Attribute-Based Encryption

3.1 Algorithms

A Ciphertext-Policy Attribute-Based Encryption scheme consists of the following four PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \to (pp, msk)$**:** The Setup algorithm takes the security parameter $\lambda \in \mathbb{N}$ encoded in unary and outputs the public parameters pp and the master secret key msk. We assume that the public parameters contain a description of the attribute universe $\mathcal{U}$. [3]

- $\mathsf{KeyGen}(1^\lambda, pp, msk, \mathcal{S}) \to sk$**:** The key generation algorithm takes as inputs the public parameters pp, the master secret key msk and a set of attributes $\mathcal{S} \subseteq \mathcal{U}$. The security parameter is included in the inputs to ensure that it is polynomial time in λ. The algorithm generates a secret key corresponding to $\mathcal{S}$.

- $\mathsf{Encrypt}(1^\lambda, pp, m, \mathbb{A}) \to ct$**:** The encryption algorithm takes as inputs the public parameters pp, a plaintext message m, and an access structure $\mathbb{A}$ on $\mathcal{U}$. It outputs the ciphertext ct.

- $\mathsf{Decrypt}(1^\lambda, pp, sk, ct) \to m$**:** The decryption algorithm takes as inputs the public parameters pp, a secret key sk, and a ciphertext ct. It outputs the plaintext m.

[3] In previous CP-ABE constructions the attribute universe $\mathcal{U}$ was one of the arguments of the Setup algorithm. In the large universe case, the attribute universe depends only on the size of the underlying group $\mathbb{G}$, which depends on the security parameter λ and the group generation algorithm.

Correctness: We require that a CP-ABE scheme is correct, i.e the decryption algorithm correctly decrypts a ciphertext of an access structure $\mathbb{A}$ with a secret key on $\mathcal{S}$, when $\mathcal{S}$ is an authorized set of $\mathbb{A}$. Formally:

Definition 3.1. *A CP-ABE scheme is* correct *when for all messages m, and all attribute sets $\mathcal{S}$ and access structures $\mathbb{A}$ with $\mathcal{S} \in \mathbb{A}$ (i.e. for $\mathcal{S}$ authorized):*

$$\Pr\left[\mathsf{Decrypt}(1^\lambda, pp, sk, ct) \neq m \;\middle|\; \begin{array}{c} (pp, msk) \leftarrow \mathsf{Setup}(1^\lambda) \\ sk \leftarrow \mathsf{KeyGen}(1^\lambda, pp, msk, \mathcal{S}) \\ ct \leftarrow \mathsf{Encrypt}(1^\lambda, pp, m, \mathbb{A}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over all random coins of all algorithms.

Remarks: According to the above definition of CP-ABE, all algorithms except Setup take as input the security parameter and the public parameters. Since these are meant to be publicly available to anyone, we will omit writing them explicitly in the construction; for example we will write $\mathsf{KeyGen}(msk, \mathcal{S})$ meaning $\mathsf{KeyGen}(1^\lambda, pp, msk, \mathcal{S})$.

Also, for the encryption algorithm $\mathsf{Encrypt}$, we will assume that its last input is not an access structure $\mathbb{A}$, but the corresponding policy (M, ρ). Of course the transformation from monotone Boolean formulas to policies is included in the running times of the implementations in section 5.

3.2 CP-ABE Selective Security

In this section we present the definition of selective security for CP-ABE schemes. This is described by a game between a challenger and an attacker and is parameterized by the security parameter $\lambda \in \mathbb{N}$. The phases of the game are the following:

- **Initialization:** In this phase the attacker declares the challenge access structure $\mathbb{A}^*$, which he will try to attack, and sends it to the challenger.
- **Setup:** Here the challenger calls the $\mathsf{Setup}(1^\lambda)$ algorithm and sends the public parameters pp to the attacker.
- **Query Phase 1:** In this phase the attacker can adaptively ask for secret keys for the sets of attributes $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{Q_1}$. For each $\mathcal{S}_i$ the challenger calls $\mathsf{KeyGen}(msk, \mathcal{S}_i) \rightarrow sk_i$ and sends sk_i to the attacker. The restriction that has to be satisfied for each query is that none of the queried sets satisfies the challenge access structure, i.e. $\forall i \in [Q_1] : \mathcal{S}_i \notin \mathbb{A}^*$.
- **Challenge:** The attacker declares two equal-length plaintexts m_0 and m_1 and sends them to the challenger. He flips a random coin $b \in \{0, 1\}$ and calls $\mathsf{Encrypt}(m_b, \mathbb{A}^*) \rightarrow ct$. He sends ct to the attacker.
- **Query Phase 2:** This the same as query phase 1. The attacker asks for the secret key for the sets $\mathcal{S}_{Q_1+1}, \mathcal{S}_{Q_1+2}, \ldots, \mathcal{S}_Q$, for which the same restriction holds: $\forall i \in [Q] : \mathcal{S}_i \notin \mathbb{A}^*$.
- **Guess:** The attacker outputs his guess $b' \in \{0, 1\}$ for b.

Definition 3.2. *A CP-ABE scheme is* selectively secure *if all PPT attackers have at most a negligible advantage in λ in the above security game, where the advantage of an attacker is defined as $\mathsf{Adv} = \Pr[b' = b] - 1/2$.*

4 Our Large Universe CP-ABE

In this section we present our large universe CP-ABE construction. The public parameters consist of the six group elements $(g, u, h, w, v, e(g, g)^\alpha)$, which intuitively are utilized in two separate "layers" to achieve secure large universe CP-ABE. In the "attribute layer", the u, h terms provide a Boneh-Boyen-style [7] hash function $(u^A h)$, while in the "secret sharing layer" the w term holds the secret randomness r during key generation and the shares of the secret randomness s during encryption. The v term is used to "bind" the two layers together. The g and $e(g, g)^\alpha$ terms are used to introduce the master secret key functionality and allow correct decryption.

4.1 Construction

Our scheme consists of the following four algorithms:

• $\mathsf{Setup}(1^\lambda) \to (pp, msk)$**:** The setup algorithm calls the group generator algorithm $\mathcal{G}(1^\lambda)$ and gets the descriptions of the groups and the bilinear mapping $D = (p, \mathbb{G}, \mathbb{G}_T, e)$, where p is the prime order of the groups $\mathbb{G}$ and $\mathbb{G}_T$. The attribute universe is $\mathcal{U} = \mathbb{Z}_p$.

Then the algorithm picks the random terms $g, u, h, w, v \xleftarrow{\$} \mathbb{G}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$. It outputs

$$pp = (D, g, u, h, w, v, e(g,g)^\alpha) \qquad msk = (\alpha)$$

• $\mathsf{KeyGen}(msk, \mathcal{S} = \{A_1, A_2, \ldots, A_k\} \subseteq \mathbb{Z}_p) \to sk$**:** Initially, the key generation algorithm picks $k+1$ random exponents $r, r_1, r_2, \ldots, r_k \xleftarrow{\$} \mathbb{Z}_p$. Then it computes $K_0 = g^\alpha w^r$, $K_1 = g^r$, and for every $\tau \in [k]$

$$K_{\tau,2} = g^{r_\tau} \text{ and } K_{\tau,3} = (u^{A_\tau} h)^{r_\tau} v^{-r}$$

The secret key output is $sk = (\mathcal{S}, K_0, K_1, \{K_{\tau,2}, K_{\tau,3}\}_{\tau \in [k]})$.

• $\mathsf{Encrypt}(m \in \mathbb{G}_T, (M, \rho) \in (\mathbb{Z}_p^{\ell \times n}, \mathcal{F}([\ell] \to \mathbb{Z}_p))) \to ct$**:** The encryption algorithm takes the plaintext message m and picks $\vec{y} = (s, y_2, \ldots, y_n)^\perp \xleftarrow{\$} \mathbb{Z}_p^{n \times 1}$. In the terminology of section 2.2, s is the random secret to be shared among the shares. The vector of the shares is

$$\vec{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_\ell)^\perp = M\vec{y}$$

It then picks ℓ random exponents $t_1, t_2, \ldots, t_\ell \xleftarrow{\$} \mathbb{Z}_p$ and calculates $C = m \cdot e(g,g)^{\alpha s}$, $C_0 = g^s$, and for every $\tau \in [\ell]$

$$C_{\tau,1} = w^{\lambda_\tau} v^{t_\tau}, \quad C_{\tau,2} = (u^{\rho(\tau)} h)^{-t_\tau} \text{ and } C_{\tau,3} = g^{t_\tau}$$

The ciphertext output is $ct = ((M, \rho), C, C_0, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau \in [\ell]})$.

• $\mathsf{Decrypt}\,(sk, ct) \to m$**:** Firstly, the decryption algorithm calculates the set of rows in M that provide a share to attributes in $\mathcal{S}$, i.e. $I = \{i : \rho(i) \in S\}$. Then it computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i M_i = (1, 0, \ldots, 0)$, where M_i is the i-th row of the matrix M. According to the discussion in section 2.2, these constants exist if the set $\mathcal{S}$ is an authorized set of the policy.

Then it calculates

$$B = \frac{e(C_0, K_0)}{\prod_{i \in I} \left(e(C_{i,1}, K_1) e(C_{i,2}, K_{\tau,2}) e(C_{i,3}, K_{\tau,3})\right)^{\omega_i}}$$

where τ is the index of the attribute $\rho(i)$ in $\mathcal{S}$ (it depends on i). The algorithm outputs $m = C/B$.

Correctness: If the attribute set $\mathcal{S}$ of the secret key is authorized, we have that $\sum_{i \in I} \omega_i \lambda_i = s$. Therefore:

$$\begin{aligned} B &= \frac{e(g,g)^{\alpha s} e(g,w)^{rs}}{\prod_{i \in I} e(g,w)^{r\omega_i\lambda_i} e(g,v)^{rt_i\omega_i} e(g, u^{\rho(i)}h)^{-r_\tau t_i \omega_i} e(g, u^{\rho(i)}h)^{r_\tau t_i \omega_i} e(g,v)^{-rt_i\omega_i}} \\ &= \frac{e(g,g)^{\alpha s} e(g,w)^{rs}}{e(g,w)^{r \sum_{i \in I} \omega_i \lambda_i}} = e(g,g)^{\alpha s} \end{aligned}$$

4.2 Proof of Selective Security

We will prove the following theorem regarding the selective security of our CP-ABE scheme:

Theorem 4.1. *If the q-1 assumption holds then all PPT adversaries with a challenge matrix of size $\ell \times n$, where $\ell, n \leq q$, have a negligible advantage in selectively breaking our scheme.*

Proof. To prove the theorem we will assume that there exists a PPT attacker $\mathcal{A}$ with a challenge matrix that satisfies the restriction, which has a non negligible advantage $\mathsf{Adv}_{\mathcal{A}}$ in selectively breaking our scheme. Using this attacker we will build a PPT simulator $\mathcal{B}$ that attacks the q-1 assumption with a non negligible advantage.

Initialization: $\mathcal{B}$ receives the given terms from the assumption and a challenge policy (M^*, ρ^*) from $\mathcal{A}$. We have that M^* is an $\ell \times n$ matrix, where $\ell, n \leq q$, and $\rho^* \in \mathcal{F}([\ell] \to \mathbb{Z}_p)$.

Setup: The simulator $\mathcal{B}$ has to provide $\mathcal{A}$ the public parameters of the system. In order to do that it implicitly sets the master secret key of the scheme to be $\alpha = a^{q+1} + \tilde{\alpha}$, where a, q are set in the assumption and $\tilde{\alpha} \xleftarrow{\$} \mathbb{Z}_p$ is a known to $\mathcal{B}$ random exponent. Notice that this way α is correctly distributed and a is information-theoretically hidden from $\mathcal{A}$. Then $\mathcal{B}$ picks the random exponents $\tilde{v}, \tilde{u}, \tilde{h} \xleftarrow{\$} \mathbb{Z}_p$ and using the assumption gives to $\mathcal{A}$ the following public parameters:

$$
\begin{array}{ll}
g = g & w = g^a \\
v = g^{\tilde{v}} \cdot \prod_{(j,k)\in[\ell,n]} \left(g^{a^k/b_j}\right)^{M^*_{j,k}} & u = g^{\tilde{u}} \cdot \prod_{(j,k)\in[\ell,n]} \left(g^{a^k/b_j^2}\right)^{M^*_{j,k}} \\
h = g^{\tilde{h}} \cdot \prod_{(j,k)\in[\ell,n]} \left(g^{a^k/b_j^2}\right)^{-\rho^*(j)M^*_{j,k}} & e(g,g)^{\alpha} = e(g^a, g^{a^q}) \cdot e(g,g)^{\tilde{\alpha}}
\end{array}
$$

The term w is properly distributed in $\mathcal{A}$'s view because the the term $e(g,g)^\alpha$ hides the exponent a information-theoretically. The terms v, u, h are also properly distributed due to $\tilde{v}, \tilde{u}, \tilde{h}$, respectively. Notice that all terms can be calculated by the simulator using suitable terms from the assumption and the challenge policy given by $\mathcal{A}$.

As one can see, the "attribute layer", which consists of the terms u, h, is made up of terms whose exponents have b_i^2 in the denominator, the "binder term" v has b_i, and the "secret sharing layer" w has only one power of a. This scaling of the powers of b_i will allow our simulator to properly simulate all terms.

Query phases 1 and 2: Now the simulator has to produce secret keys for non authorized sets of attributes requested by $\mathcal{A}$. In both phases the treatment is the same. We describe here the way $\mathcal{B}$ works in order to create a key for an attribute set $\mathcal{S} = \left\{A_1, A_2, \ldots, A_{|\mathcal{S}|}\right\}$ received by $\mathcal{A}$.

Since $\mathcal{S}$ is non authorized for (M^*, ρ^*), there exists a vector $\vec{w} = (w_1, w_2, \ldots, w_n)^\perp \in \mathbb{Z}_p^n$ such that $w_1 = -1$ and $\langle M_i^*, \vec{w}\rangle = 0$ for all $i \in I = \{i | i \in [\ell] \wedge \rho^*(i) \in \mathcal{S}\}$ (c.f. section 2.2). The simulator calculates $\vec{w}$ using linear algebra. Then it picks $\tilde{r} \xleftarrow{\$} \mathbb{Z}_p$ and implicitly sets

$$
r = \tilde{r} + w_1 a^q + w_2 a^{q-1} + \ldots + w_n a^{q+1-n} = \tilde{r} + \sum_{i\in[n]} w_i a^{q+1-i}
$$

This is properly distributed due to $\tilde{r}$. Then using the suitable terms from the assumption it calculates:

$$
\begin{aligned}
K_0 &= g^\alpha w^r = g^{a^{q+1}} g^{\tilde{\alpha}} g^{a\tilde{r}} \prod_{i\in[n]} g^{w_i a^{q+2-i}} = g^{\tilde{\alpha}} \left(g^a\right)^{\tilde{r}} \prod_{i=2}^{n} \left(g^{a^{q+2-i}}\right)^{w_i} \\
K_1 &= g^r = g^{\tilde{r}} \prod_{i\in[n]} \left(g^{a^{q+1-i}}\right)^{w_i}
\end{aligned}
$$

Additionally, for all $\tau \in [|\mathcal{S}|]$ it has to compute the terms $K_{\tau,2} = g^{r_\tau}$ and $K_{\tau,3} = (u^{A_\tau} h)^{r_\tau} v^{-r}$. The common part v^{-r} for these terms is the following:

$$\begin{aligned}
v^{-r} &= v^{-\tilde{r}} \left(g^{\tilde{v}} \prod_{(j,k)\in[\ell,n]} g^{a^k M^*_{j,k}/b_j} \right)^{-\sum_{i\in[n]} w_i a^{q+1-i}} \\
&= v^{-\tilde{r}} \prod_{i\in[n]} \left(g^{a^{q+1-i}}\right)^{-\tilde{v}w_i} \cdot \prod_{(i,j,k)\in[n,\ell,n]} g^{-w_i M^*_{j,k} a^{q+1+k-i}/b_j} \\
&= \underbrace{v^{-\tilde{r}} \prod_{i\in[n]} \left(g^{a^{q+1-i}}\right)^{-\tilde{v}w_i} \cdot \prod_{\substack{(i,j,k)\in[n,\ell,n] \\ i\neq k}} \left(g^{a^{q+1+k-i}/b_j}\right)^{-w_i M^*_{j,k}}}_{\Phi} \cdot \prod_{(i,j)\in[n,\ell]} g^{-w_i M^*_{j,i} a^{q+1}/b_j} \\
&= \Phi \cdot \prod_{j\in[\ell]} g^{-\langle \vec{w}, M^*_j \rangle a^{q+1}/b_j} \\
&= \Phi \cdot \prod_{\substack{j\in[\ell] \\ \rho^*(j)\notin S}} g^{-\langle \vec{w}, M^*_j \rangle a^{q+1}/b_j}
\end{aligned}$$

The Φ part can be calculated by the simulator using the assumption, while the second part has to be canceled by the $(u^{A_\tau} h)^{r_\tau}$ part. So for every attribute $A_\tau \in S$ the simulator sets implicitly

$$\begin{aligned}
r_\tau &= \tilde{r}_\tau + r \cdot \sum_{\substack{i'\in[\ell] \\ \rho^*(i')\notin S}} \frac{b_{i'}}{A_\tau - \rho^*(i')} \\
&= \tilde{r}_\tau + \tilde{r} \cdot \sum_{\substack{i'\in[\ell] \\ \rho^*(i')\notin S}} \frac{b_{i'}}{A_\tau - \rho^*(i')} + \sum_{\substack{(i,i')\in[n,\ell] \\ \rho^*(i')\notin S}} \frac{w_i b_{i'} a^{q+1-i}}{A_\tau - \rho^*(i')}
\end{aligned}$$

where $\tilde{r}_\tau \xleftarrow{\$} \mathbb{Z}_p$ and therefore r_τ is properly distributed. The use of the b_i's in the numerators of the fractions is explained by the "layer" intuition presented before. Namely, these b_i will cancel with the b_i^2 denominators in the "attribute layer" and provide a cancellation for the unknown part of v^{-r}.

Also, notice that r_τ is well-defined only for attributes in the specific unauthorized set $\mathcal{S}$ or unrelated attributes (outside the policy), since the sum is over the i' such that $\rho^*(i') \notin \mathcal{S}$. Therefore, for all $A_\tau \in \mathcal{S}$ or $A_\tau \notin \rho^*([\ell])$, the denominators $A_\tau - \rho^*(i')$ are non zero. If the simulator tries to include more attributes of the policy in the key (and possibly make a key for an authorized set), he would have to divide by zero (c.f. Figure 1).

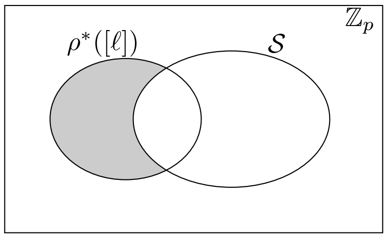

Figure 1: The simulator can not create the components for attributes in the gray area.

Therefore the first part of $K_{\tau,3} = (u^{A_\tau} h)^{r_\tau} v^{-r}$ is:

$$
\begin{aligned}
(u^{A_\tau}h)^{r_\tau} &= (u^{A_\tau}h)^{\tilde{r}_\tau} \cdot \left(g^{\tilde{u}A_\tau + \tilde{h}} \prod_{(j,k)\in[\ell,n]} g^{(A_\tau - \rho^*(j))M^*_{j,k}a^k/b_j^2} \right)^{\tilde{r}\cdot \sum_{i'\in[\ell],\rho^*(i')\notin S} \frac{b_{i'}}{A_\tau - \rho^*(i')}} \\
&\quad \cdot \left(g^{\tilde{u}A_\tau + \tilde{h}} \prod_{(j,k)\in[\ell,n]} g^{(A_\tau - \rho^*(j))M^*_{j,k}a^k/b_j^2} \right)^{\sum_{(i,i')\in[n,\ell],\rho^*(i')\notin S} \frac{w_i b_{i'} a^{q+1-i}}{A_\tau - \rho^*(i')}} \\
&= (u^{A_\tau}h)^{\tilde{r}_\tau} \cdot (K_{\tau,2}/g^{\tilde{r}_\tau})^{\tilde{u}A_\tau + \tilde{h}} \cdot \prod_{\substack{(i',j,k)\in[\ell,\ell,n] \\ \rho^*(i')\notin S}} g^{\tilde{r}(A_\tau - \rho^*(j))M^*_{j,k} b_{i'} a^k/(A_\tau - \rho^*(i'))b_j^2} \\
&\quad \cdot \prod_{\substack{(i,i',j,k)\in[n,\ell,\ell,n] \\ \rho^*(i')\notin S}} g^{(A_\tau - \rho^*(j))w_i M^*_{j,k} b_{i'} a^{q+1+k-i}/(A_\tau - \rho^*(i'))b_j^2} \\
&= \Psi \cdot \prod_{\substack{(i,j)\in[n,\ell] \\ \rho^*(j)\notin S}} g^{(A_\tau - \rho^*(j))w_i M^*_{j,i} b_j a^{q+1+i-i}/(A_\tau - \rho^*(j))b_j^2} \\
&= \Psi \cdot \prod_{\substack{j\in[\ell] \\ \rho^*(j)\notin S}} g^{\langle \vec{w}, M^*_j \rangle a^{q+1}/b_j}
\end{aligned}
$$

$$
\text{Where } \Psi = (u^{A_\tau}h)^{\tilde{r}_\tau} \cdot (K_{\tau,2}/g^{\tilde{r}_\tau})^{\tilde{u}A_\tau + \tilde{h}} \cdot \prod_{\substack{(i',j,k)\in[\ell,\ell,n] \\ \rho^*(i')\notin S}} \left(g^{b_{i'}a^k/b_j^2} \right)^{\tilde{r}(A_\tau - \rho^*(j))M^*_{j,k}/(A_\tau - \rho^*(i'))}
$$

$$
\cdot \prod_{\substack{(i,i',j,k)\in[n,\ell,\ell,n] \\ \rho^*(i')\notin S, (j\neq i' \vee i\neq k)}} \left(g^{b_{i'}a^{q+1+k-i}/b_j^2} \right)^{(A_\tau - \rho^*(j))w_i M^*_{j,k}/(A_\tau - \rho^*(i'))}
$$

$$
\text{and } K_{\tau,2} = g^{r_\tau} = g^{\tilde{r}_\tau} \cdot \prod_{\substack{i'\in[\ell] \\ \rho^*(i')\notin S}} \left(g^{b_{i'}} \right)^{\tilde{r}/(A_\tau - \rho^*(i'))} \cdot \prod_{\substack{(i,i')\in[n,\ell] \\ \rho^*(i')\notin S}} \left(g^{b_{i'}a^{q+1-i}} \right)^{w_i/(A_\tau - \rho^*(i'))}
$$

The Ψ and $K_{\tau,2}$ terms can be calculated using the suitable terms of our assumption[4]. The second part of $(u^{A_\tau}h)^{r_\tau}$ cancels exactly with the problematic part of v^{-r}. Therefore the simulator can calculate $K_{\tau,2}$ and $K_{\tau,3}$ for all $A_\tau \in \mathcal{S}$ and hand over the secret key $sk = (\mathcal{S}, K_0, K_1, \{K_{\tau,2}, K_{\tau,3}\}_{\tau\in[|\mathcal{S}|]})$ to the attacker $\mathcal{A}$.

Challenge: The attacker will output a pair of messages (m_0, m_1) of the same length. In this phase the simulator flips a random coin $b \xleftarrow{\$} \{0,1\}$ and constructs

$$
C = m_b \cdot T \cdot e(g, g^s)^{\tilde{\alpha}} \qquad \text{and} \qquad C_0 = g^s
$$

where T is the challenge term and g^s the corresponding term of the assumption.

The simulator sets implicitly $\vec{y} = \left(s, sa + \tilde{y}_2, sa^2 + \tilde{y}_3, \ldots, sa^{n-1} + \tilde{y}_n\right)^\perp$, where $\tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n \xleftarrow{\$} \mathbb{Z}_p$. We see that the secret s and the vector $\vec{y}$ are properly distributed, since s was information theoretically hidden from $\mathcal{A}$ and the $\tilde{y}_i$'s are picked uniformly at random. As a result, since $\vec{\lambda} = M^*\vec{y}$ we have that

$$
\lambda_\tau = \sum_{i\in[n]} M^*_{\tau,i} s a^{i-1} + \sum_{i=2}^{n} M^*_{\tau,i}\tilde{y}_i = \sum_{i\in[n]} M^*_{\tau,i} s a^{i-1} + \tilde{\lambda}_\tau
$$

[4]Notice that for the products of Ψ we can have $j = i'$, but in that case the power of a is different than $q+1$. So the simulator can use the g^{a^i/b_j} terms.

for each row $\tau \in [\ell]$. Notice that the terms $\tilde{\lambda}_\tau = \sum_{i=2}^{n} M^*_{\tau,i}\tilde{y}_i$ are known to the simulator. For each row the simulator $\mathcal{B}$ sets implicitly $t_\tau = -sb_\tau$. This is properly distributed as well, because the b_i's are information theoretically hidden from the attacker. Using the above, $\mathcal{B}$ calculates:

$$\begin{aligned}
C_{\tau,1} &= w^{\lambda_\tau} v^{t_\tau} = w^{\tilde{\lambda}_\tau} \cdot \prod_{i\in[n]} g^{M^*_{\tau,i} sa^i} \cdot \left(g^{sb_\tau}\right)^{-\tilde{v}} \cdot \prod_{(j,k)\in[\ell,n]} g^{-M^*_{j,k}a^k sb_\tau/b_j} = \\
&= w^{\tilde{\lambda}_\tau} \cdot \left(g^{sb_\tau}\right)^{-\tilde{v}} \cdot \prod_{i\in[n]} g^{M^*_{\tau,i} sa^i} \cdot \prod_{k\in[n]} g^{-M^*_{\tau,k}a^k sb_\tau/b_\tau} \cdot \prod_{\substack{(j,k)\in[\ell,n] \\ j\neq\tau}} g^{-M^*_{j,k}a^k sb_\tau/b_j} = \\
&= w^{\tilde{\lambda}_\tau} \cdot \left(g^{sb_\tau}\right)^{-\tilde{v}} \cdot \prod_{\substack{(j,k)\in[\ell,n] \\ j\neq\tau}} \left(g^{sa^k b_\tau/b_j}\right)^{-M^*_{j,k}} \\
C_{\tau,2} &= \left(u^{\rho^*(\tau)} h\right)^{-t_\tau} = \left(g^{sb_\tau}\right)^{-(\tilde{u}\rho^*(\tau)+\tilde{h})} \cdot \left(\prod_{(j,k)\in[\ell,n]} g^{(\rho^*(\tau)-\rho^*(j))M^*_{j,k}a^k/b_j^2}\right)^{-sb_\tau} \\
&= \left(g^{sb_\tau}\right)^{-(\tilde{u}\rho^*(\tau)+\tilde{h})} \cdot \prod_{\substack{(j,k)\in[\ell,n] \\ j\neq\tau}} \left(g^{sa^k b_\tau/b_j^2}\right)^{-(\rho^*(\tau)-\rho^*(j))M^*_{j,k}} \\
C_{\tau,3} &= g^{t_\tau} = \left(g^{sb_\tau}\right)^{-1}
\end{aligned}$$

Notice that by using $t_\tau = -sb_\tau$ we "raised" the exponents of the "binder" term v so that they cancel with the unknown powers of w^{λ_τ}. Therefore, the simulator hands over the ciphertext $ct = ((M^*, \rho^*), C, C_0, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau\in[\ell]})$ to the attacker $\mathcal{A}$.

Guess: After the query phase 2, where the simulator creates the secret keys as described above, the attacker outputs a guess b' for the challenge bit. If $b' = b$ the simulator outputs 0, i.e. it claims that the challenge term is $T = e(g,g)^{sa^{q+1}}$. Otherwise, it outputs 1.

If $T = e(g,g)^{sa^{q+1}}$ then $\mathcal{A}$ played the proper security game, because $C = m_b \cdot T \cdot e(g, g^s)^{\tilde{\alpha}} = m_b \cdot e(g,g)^{\alpha s}$. On the other hand, if T is a random term of $\mathbb{G}_T$ then all information about the message m_b is lost in the challenge ciphertext. Therefore the advantage of $\mathcal{A}$ is exactly 0. As a result if $\mathcal{A}$ breaks the proper security game with a non negligible advantage, then $\mathcal{B}$ has a non negligible advantage in breaking the q-1 assumption. ■

5 Implementation and Evaluation

Implementation Details We implemented our schemes in Charm [1]; a framework developed to facilitate the rapid prototyping of cryptographic schemes and protocols. It is based on the Python language which allows the programmer to write code similar to the theoretical implementations. However, the routines that implement the dominant group operations use the PBC library [27] (written natively in C) and the time overhead imposed by the use of Python is usually less than 1%. Charm also provides routines for applying and using LSSS schemes needed for Attribute-Based systems. All Charm routines use formally asymmetric groups (although the underlining groups might be symmetric) and therefore we translated our schemes to the asymmetric setting. Namely, we have three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing e is a function from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$. The assumptions and the security proofs can be translated to the asymmetric setting in a generic way. For more information on Charm we refer the reader to [12, 1]. The source code of our implementations can be found in [41][5]. All our benchmarks were executed on a dual core Intel® Xeon® CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04 and Python3.2.3.

[5]For submission: We will make the code available to the reviewers upon request.

Scheme	Algorithm	$\mathbb{Z}_p$		$\mathbb{G}_1$		$\mathbb{G}_2$		$\mathbb{G}_T$		Pairings
		Add.	Mul.	Op.	Exp.	Op.	Exp.	Op.	Exp.	
KP-ABE from App.C	Setup	0	0	0	0	0	0	0	1	1
	KeyGen	12	12	8	16	0	4	0	0	0
	Encrypt	0	0	8	9	0	5	1	1	0
	Decrypt	4	10	0	0	0	0	7	2	6
KP-ABE from [22]	Setup	1395	1657	0	60	0	80	0	2	1
	KeyGen	24	104	0	0	200	240	0	0	0
	Encrypt	0	80	170	180	0	0	2	2	0
	Decrypt	4	10	0	0	0	0	21	20	20
CP-ABE from Sec.4	Setup	0	0	0	1	0	0	0	1	1
	KeyGen	0	0	9	10	0	5	0	0	0
	Encrypt	12	12	8	16	0	5	1	1	0
	Decrypt	4	10	0	0	0	0	8	2	7

Table 1: Group operation benchmarks. Add. denotes the number of additions and subtractions in the $\mathbb{Z}_p$ group, while Mul. the number of multiplications and divisions. Op. and Exp. denote the numbers of group operations and exponentiations, respectively, in groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$. Notice the large number of operations in $\mathbb{Z}_p$ for the Setup of the second scheme, due to the creation of dual vector spaces of dimension 10 (c.f. [22]). We mention that the last scheme is of different type and *should not be compared* to the other two.

We implemented our two ABE schemes and the prime order KP-ABE construction from [22]. Actually, in that work a large universe prime order HIBE is provided, but the transformation to KP-ABE is straightforward by substituting in the key generation algorithm the additive shares of the secrets with the LSSS shares and the identities with the attributes $\rho(\tau)$. This modified construction is the one we used for comparison to our KP-ABE scheme. At the time of writing this paper only this large universe prime order KP-ABE construction is known. Also, to the best of our knowledge, no large universe CP-ABE constructions in the standard model have been published.

Group Operations (Absolute) In table 1 we show the number of operations in the respective groups for each algorithm of the schemes as counted by the Charm benchmarking utility. The group operations refer to the number of arithmetic operations in $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. These will be the same for all subsequent benchmarks. We notice here that we tried to implement our algorithms so that more operations are executed in the $\mathbb{G}_1$ group than in the $\mathbb{G}_2$ and that encryption consists mainly of operations in $\mathbb{G}_1$, compared to key generation. The reason is that the time taken to execute them in the $\mathbb{G}_1$ group is considerably smaller than $\mathbb{G}_2$ in specific asymmetric groups such as the "MNT224" elliptic curve group. This group was created by Miyaji, Nakabayashi and Takano [29], where the base field size is 224 bits (see [27] for more information).

For the KP-ABE setting the number of group operations during the key generation, encryption, and decryption calls depends linearly on the number of rows in the policy, on the size of the attribute set and the number of rows that are used during decryption, respectively. In the CP-ABE setting the key generation time grows linearly with the size of the attribute set and the encryption time with the number of rows in the policy. In table 1 and all our time benchmarks, we showcase a small example where the policies used are of the form $(A \vee B) \wedge (C \vee D)$ and the attribute sets of the form $\{A, C, E, F\}$, where A, B, C, D, E, F are 6 different attributes. Notice that the attribute set always satisfies the policy; therefore, we measure successful decryptions that utilize 2 rows of the policy (the "A" and "C" row).

Group Operations (Asymptotic) In table 2, we demonstrate the asymptotic growth of the non constant algorithms of the schemes implemented. Some constant factors might not correspond exactly to the factors that can be derived from schemes in section 4 and appendix C, because certain optimizations have been applied so that common parts are only computed once (see [41] for more details). The results of table 1 can be obtained by setting $k = 4$, $m = 4$ and $n = 2$.

Scheme	Algorithm	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	Pairings
KP-ABE from App.C	KeyGen	$4k$	k	0	0
	Encrypt	$2m+1$	$m+1$	1	0
	Decrypt	0	0	n	$3n$
KP-ABE from [22]	KeyGen	0	$60k$	0	0
	Encrypt	$40m+20$	0	2	0
	Decrypt	0	0	$10n$	$10n$
CP-ABE from Sec.4	KeyGen	$2m+2$	$m+1$	0	0
	Encrypt	$4k$	$k+1$	1	0
	Decrypt	0	0	n	$3n+1$

Table 2: Asymptotic growth of the *exponentations* in the three groups and the pairings. These are the dominant operations in each call. The Setup algorithms have constant size and they are excluded. The k, m, n parameters denote the number of rows of the policy, the size of the attribute set, and the rows utilized during decryption, respectively. Notice that the parameters k and m have switched algorithms in the CP-ABE case. Thus, the last scheme *should not be compared* to the other two.

Time Benchmarks In table 3 we present time benchmarks in different elliptic curve groups by averaging the times of respective algorithms over 10 different instantiations of each scheme. We note that in all of these instantiations the numbers of operations in each group remain the same as the ones shown in table 1, since we used exactly the same attribute sets and policies. One interesting observation is the huge difference between operations in the $\mathbb{G}_1$ and $\mathbb{G}_2$ groups in the asymmetric "MNT" groups. This is most prominent in the [22]-KP-ABE scheme.

Conclusion Regarding the comparison between our KP-ABE scheme and the only known KP-ABE scheme on prime order groups, we notice the big gap between the asymptotic bounds as well as the timing benchmarks. This is due to the fact that Dual Vector Spaces of high dimension (in this case 10) are utilized, which increase the number of group operations by big factors.

Regarding the practicality, in general, of both our schemes we notice that the KeyGen, Encrypt, and Decrypt times of our algorithms are relatively small. They are all under 100ms, with the exception of the super singular 1024-bit curve. Even for this curve the times for each algorithm are under the 700 msec mark. Although one would expect that as the policies and the attributes sets grow bigger these times will increase, the additional overhead will grow only linearly. Thus we believe that the two constructions constitute the most practical implementations of large universe ABE, secure in the standard model.

Curve	Scheme	Setup	KeyGen	Encrypt	Decrypt
"MNT159"	KP-ABE from App.C	21.0	43.8	44.0	36.9
	KP-ABE from [22]	688.2	1671.5	168.5	125.6
	CP-ABE from Sec.4	23.3	43.5	53.6	41.9
"MNT201"	KP-ABE from App.C	27.8	59.1	59.2	50.3
	KP-ABE from [22]	924.4	2294.2	239.7	173.5
	CP-ABE from Sec.4	31.0	58.5	71.5	57.5
"MNT224"	KP-ABE from App.C	34.0	73.6	73.8	61.6
	KP-ABE from [22]	1146.1	2878.6	302.3	216.6
	CP-ABE from Sec.4	37.8	73.0	87.9	71.6
"SS512"	KP-ABE from App.C	18.9	47.9	30.2	15.1
	KP-ABE from [22]	442.8	640.0	480.8	43.8
	CP-ABE from Sec.4	24.7	32.6	52.0	16.3
"SS1024"	KP-ABE from App.C	71.2	622.6	393.4	320.8
	KP-ABE from [22]	5518.5	9238.7	6931.9	1088.7
	CP-ABE from Sec.4	115.5	427.8	665.6	373.0

Table 3: These results are average running times in milliseconds over 10 different instantiations of each scheme. "MNT" are the Miyaji, Nakabayashi, Takano curves (asymmetric pairing groups), while "SS" are super singular curves (symmetric pairing groups). The number after the type of the curve denotes the size of the base field in bits. KeyGen and Encrypt are called with attribute sets and policies of size 4, while Decrypt with common attribute sets of size 2. As before, we note that the CP-ABE results *should not be compared* to the other results.

References

[1] Joseph A. Akinyele, Matthew Green, and Avi Rubin. Charm: A framework for rapidly prototyping cryptosystems. Cryptology ePrint Archive, Report 2011/617, 2011. `http://eprint.iacr.org/`.

[2] Sattam S. Al-Riyami, John Malone-Lee, and Nigel P. Smart. Escrow-free encryption supporting cryptographic workflow. *Int. J. Inf. Sec.*, 5(4):217–229, 2006.

[3] Walid Bagga, Refik Molva, and Stefano Crosta. Policy-based encryption schemes from bilinear pairings. In *ASIACCS*, page 368, 2006.

[4] Manuel Barbosa and Pooya Farshim. Secure cryptographic workflow in the standard model. In *INDOCRYPT*, pages 379–393, 2006.

[5] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Dept. of Computer Science, Technion, 1996.

[6] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[7] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001.

[9] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[10] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[11] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons. Concealing complex policies with hidden credentials. In *ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[12] Charm. `http://www.charm-crypto.com`.

[13] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

[14] Melissa Chase and Sherman S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[15] Ling Cheung and Calvin C. Newport. Provably secure ciphertext policy ABE. In *ACM Conference on Computer and Communications Security*, pages 456–465, 2007.

[16] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[17] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[18] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[19] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP (2)*, pages 579–591, 2008.

[20] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[21] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[22] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.

[23] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[24] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.

[25] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In *EUROCRYPT*, pages 547–567, 2011.

[26] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, pages 180–198, 2012.

[27] Ben Lynn. The Stanford pairing based crypto library. `http://crypto.stanford.edu/pbc`.

[28] Gerome Miklau and Dan Suciu. Controlling access to published data using cryptography. In *VLDB*, pages 898–909, 2003.

[29] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under fr-reduction. In *ICISC*, pages 90–108, 2000.

[30] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, pages 57–74, 2008.

[31] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, pages 214–231, 2009.

[32] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[33] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.

[34] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *ACM Conference on Computer and Communications Security*, pages 99–112, 2006.

[35] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[36] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[37] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009.

[38] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *ICALP (2)*, pages 560–578, 2008.

[39] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

[40] Nigel P. Smart. Access control using pairing based cryptography. In *CT-RSA*, pages 111–121, 2003.

[41] Source code of our constructions. `www.cs.utexas.edu/~jrous/`.

[42] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[43] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, pages 53–70, 2011.

Appendices

A Assumption 2

For our KP-ABE construction we will use a q-type assumption on prime order bilinear groups, denoted by q-2, which is similar to the Decisional Bilinear Diffie-Hellman Assumption augmented with q parameters b_i. It is parameterized by a security parameter $\lambda \in \mathbb{N}$ and an integer q, polynomial in λ. We assume that there exists a group generator algorithm $\mathcal{G}(1^\lambda) \to (p, \mathbb{G}, \mathbb{G}_T, e)$ that outputs the description of the (symmetric) bilinear group of order $p = \Theta(2^\lambda)$. The generic security of the assumption is shown in appendix D. It is defined via the following game between a challenger and an attacker:

Initially the challenger calls the group generation algorithm with input the security parameter, picks a random group element $g \xleftarrow{\$} \mathbb{G}$, and $q+3$ random exponents $x, y, z, b_1, b_2, \ldots, b_q \xleftarrow{\$} \mathbb{Z}_p$. Then he sends to the attacker the group description $(p, \mathbb{G}, \mathbb{G}_T, e)$ and all of the following terms:

$$\begin{array}{ll} g, g^x, g^y, g^z, g^{(xz)^2} & \\ g^{b_i}, g^{xzb_i}, g^{xz/b_i}, g^{x^2zb_i}, g^{y/b_i^2}, g^{y^2/b_i^2} & \forall i \in [q] \\ g^{xzb_i/b_j}, g^{yb_i/b_j^2}, g^{xyzb_i/b_j^2}, g^{(xz)^2 b_i/b_j} & \forall i,j \in [q], i \neq j \end{array}$$

In the challenge phase, the challenger flips a random coin $b \xleftarrow{\$} \{0,1\}$ and if $b = 0$ it gives to the attacker the term $e(g,g)^{xyz}$. Otherwise it gives a random term $R \xleftarrow{\$} \mathbb{G}_T$. Finally the attacker outputs a guess $b' \in \{0,1\}$.

Definition A.1. *We say that the q-2 assumption holds if all PPT attackers have at most a negligible advantage in λ in the above security game, where the advantage is defined as* $\mathsf{Adv} = \Pr[b' = b] - 1/2$.

B Key-Policy Attribute-Based Encryption

B.1 Algorithms

A Key-Policy Attribute-Based Encryption scheme consists of the following four PPT algorithms:

• **$\mathsf{Setup}(1^\lambda) \to (pp, msk)$:** The Setup algorithm takes the security parameter $\lambda \in \mathbb{N}$ encoded in unary and outputs the public parameters pp and the master secret key msk. We assume that the public parameters contain a description of the attribute universe $\mathcal{U}$.

• **$\mathsf{KeyGen}(1^\lambda, pp, msk, \mathbb{A}) \to sk$:** The key generation algorithm takes as inputs the public parameters pp, the master secret key msk and an access structure $\mathbb{A}$ on $\mathcal{U}$. The algorithm generates a secret key corresponding to $\mathbb{A}$.

• **$\mathsf{Encrypt}(1^\lambda, pp, m, \mathcal{S}) \to ct$:** The encryption algorithm takes as inputs the public parameters pp, a plaintext message m, and a set of attributes $\mathcal{S} \subseteq \mathcal{U}$. It outputs the ciphertext ct.

• $\mathsf{Decrypt}(1^\lambda, pp, sk, ct) \to m$**:** The decryption algorithm takes as inputs the public parameters pp, a secret key sk, and a ciphertext ct. It outputs the plaintext m.

Correctness: We require that a KP-ABE scheme is correct, i.e the decryption algorithm correctly decrypts a ciphertext on $\mathcal{S}$ with a secret key of an access structure $\mathbb{A}$ when $\mathcal{S}$ is an authorized set of $\mathbb{A}$. Formally:

Definition B.1. *A KP-ABE scheme is* correct *when for all messages m, and all attribute sets $\mathcal{S}$ and access structures $\mathbb{A}$ with $\mathcal{S} \in \mathbb{A}$ (i.e. for $\mathcal{S}$ authorized):*

$$\Pr\left[\mathsf{Decrypt}(1^\lambda, pp, sk, ct) \neq m \;\middle|\; \begin{array}{c} (pp, msk) \leftarrow \mathsf{Setup}(1^\lambda) \\ sk \leftarrow \mathsf{KeyGen}(1^\lambda, pp, msk, \mathbb{A}) \\ ct \leftarrow \mathsf{Encrypt}(1^\lambda, pp, m, \mathcal{S}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over all random coins of all algorithms.

Remarks: Similar to the CP-ABE case we will omit the public parameters and the security parameter from the list of arguments when referring to calls of the above algorithms. Also, any access structures $\mathbb{A}$ will be represented by the corresponding policy (M, ρ).

B.2 KP-ABE Selective Security

The selective security game for KP-ABE is described by a game between a challenger and an attacker and is parameterized by the security parameter $\lambda \in \mathbb{N}$. The phases of the game are the following:

• **Initialization:** In this phase the attacker declares the challenge attribute set $\mathcal{S}^*$, which he will try to attack, and sends it to the challenger.

• **Setup:** Here the challenger calls the $\mathsf{Setup}(1^\lambda)$ algorithm and sends the public parameters pp to the attacker.

• **Query Phase 1:** In this phase the attacker can adaptively ask for secret keys for the access structures $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_{Q_1}$. For each $\mathbb{A}_i$ the challenger calls $\mathsf{KeyGen}(msk, \mathbb{A}_i) \to sk_i$ and sends sk_i to the attacker. The restriction that has to be satisfied for each query is that none of the queried policies is satisfied by the challenge attribute set, i.e. $\forall i \in [Q_1] : \mathcal{S}^* \notin \mathbb{A}_i$.

• **Challenge:** The attacker declares two equal-length plaintexts m_0 and m_1 and sends them to the challenger. He flips a random coin $b \in \{0,1\}$ and calls $\mathsf{Encrypt}(m_b, \mathcal{S}^*) \to ct$. He sends ct to the attacker.

• **Query Phase 2:** This the same as query phase 1. The attacker asks for the secret key for the access structures $\mathbb{A}_{Q_1+1}, \mathbb{A}_{Q_1+2}, \ldots, \mathbb{A}_Q$, for which the same restriction holds: $\forall i \in [Q] : \mathcal{S}^* \notin \mathbb{A}_i$.

• **Guess:** The attacker outputs his guess $b' \in \{0,1\}$ for b.

Definition B.2. *A KP-ABE scheme is* selectively secure *if all PPT attackers have at most a negligible advantage in λ in the above security game, where the advantage of an attacker is defined as* $\mathsf{Adv} = \Pr[b' = b] - 1/2$.

C Our Large Universe KP-ABE

In this section we present our large universe KP-ABE scheme. We mention here that it can be converted to an HIBE scheme using non repeating identities, "AND" policies and delegation capabilities (c.f. [25]). The intuition behind the functionality of this construction is simpler than the CP-ABE. In this setting the public parameters consist of the five terms $(g, u, h, w, e(g,g)^\alpha)$. There is one term less due to the fact that now the master secret key α is the secret to be shared during all the key generation calls. As a result the "secret sharing layer" uses the g term only and the w term is used to "bind" this layer to the u, h "attribute layer".

C.1 Construction

Our scheme consists of the following four algorithms.

• $\mathsf{Setup}(1^\lambda) \to (pp, msk)$**:** The setup algorithm calls the group generator algorithm $\mathcal{G}(1^\lambda)$ and gets the descriptions of the groups and the bilinear mapping $D = (p, \mathbb{G}, \mathbb{G}_T, e)$, where p is the prime order of the groups $\mathbb{G}$ and $\mathbb{G}_T$. The attribute universe is $\mathcal{U} = \mathbb{Z}_p$.

Then the algorithm picks the random terms $g, u, h, w \xleftarrow{\$} \mathbb{G}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$. It outputs

$$pp = (D, g, u, h, w, e(g,g)^\alpha) \qquad msk = (\alpha)$$

• $\mathsf{KeyGen}(msk, (M, \rho) \in (\mathbb{Z}_p^{\ell \times n}, \mathcal{F}([\ell] \to \mathbb{Z}_p))) \to sk$**:** Initially the algorithm picks $\vec{y} = (\alpha, y_2, \ldots, y_n)^\perp$ where $y_2, \ldots, y_n \xleftarrow{\$} \mathbb{Z}_p$. In the terminology of section 2.2, the master secret key α is the secret to be shared among the shares. The vector of the shares is

$$\vec{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_\ell)^\perp = M\vec{y}$$

It then picks ℓ random exponents $t_1, t_2, \ldots, t_\ell \xleftarrow{\$} \mathbb{Z}_p$ and for every $\tau \in [\ell]$ it computes

$$K_{\tau,0} = g^{\lambda_\tau} w^{t_\tau} \qquad K_{\tau,1} = \left(u^{\rho(\tau)} h\right)^{-t_\tau} \qquad K_{\tau,2} = g^{t_\tau}$$

The secret key is $sk = ((M, \rho), \{K_{\tau,0}, K_{\tau,1}, K_{\tau,2}\}_{\tau \in [\ell]})$.

• $\mathsf{Encrypt}(m, \mathcal{S} = \{A_1, A_2, \ldots, A_k\} \subseteq \mathbb{Z}_p) \to ct$**:** Initially, the algorithm picks $k+1$ random exponents s, r_1, r_2, …, $r_k \xleftarrow{\$} \mathbb{Z}_p$. It computes $C = m \cdot e(g,g)^{\alpha s}$, $C_0 = g^s$, and for every $\tau \in [k]$ it computes

$$C_{\tau,1} = g^{r_\tau} \qquad C_{\tau,2} = (u^{A_\tau} h)^{r_\tau} w^{-s}$$

The ciphertext is $ct = (\mathcal{S}, C, C_0, \{C_{\tau,1}, C_{\tau,2}\}_{\tau \in [k]})$.

• $\mathsf{Decrypt}\,(sk, ct) \to m$**:** The algorithm finds the set of rows in M that provide a share to attributes in $\mathcal{S}$, i.e. $I = \{i : \rho(i) \in S\}$. Then it calculate constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i M_i = (1, 0, \ldots, 0)$, where M_i is the i-th row of the matrix M. According to the discussion in section 2.2, these constants exist if the set $\mathcal{S}$ is an authorized set of the policy.

Then it calculates

$$B = \prod_{i \in I} \left(e(C_0, K_{i,0}) e(C_{\tau,1}, K_{i,1}) e(C_{\tau,2}, K_{i,2})\right)^{\omega_i}$$

where τ is the index of the attribute $\rho(i)$ in $\mathcal{S}$ (it depends on i). The algorithm outputs $m = C/B$.

Correctness: If the attribute set $\mathcal{S}$ of the ciphertext is authorized, we have that $\sum_{i \in I} \omega_i \lambda_i = \alpha$. Therefore:

$$\begin{aligned} B &= \prod_{i \in I} e(g,g)^{s\omega_i\lambda_i} e(g,w)^{st_i\omega_i} e(g, u^{\rho(i)}h)^{-r_\tau t_i \omega_i} e(g, u^{\rho(i)}h)^{r_\tau t_i \omega_i} e(g,w)^{-st_i\omega_i} \\ &= e(g,g)^{s \sum_{i \in I} \omega_i \lambda_i} = e(g,g)^{\alpha s} \end{aligned}$$

C.2 Proof of Selective Security

We will prove the following theorem regarding the selective security of our KP-ABE scheme:

Theorem C.1. *If the q-2 assumption holds, then all PPT adversaries with a challenge attribute set of size k, where $k \leq q$, have a negligible advantage in selectively breaking our scheme.*

Proof. To prove the theorem we will assume that there exists a PPT attacker $\mathcal{A}$ with a challenge attribute set that satisfies the restriction, which has a non negligible advantage $\mathsf{Adv}_{\mathcal{A}}$ in selectively breaking our scheme. Using this attacker we will build a PPT simulator $\mathcal{B}$ that attacks the q-2 assumption with a non negligible advantage.

Initialization: Initially, $\mathcal{B}$ receives the given terms from the assumption and an attribute set $\mathcal{S}^* = \{A_1^*, A_2^*, \ldots, A_k^*\} \subseteq \mathcal{U}$.

Setup: Now, the simulator $\mathcal{B}$ has to provide $\mathcal{A}$ the public parameters of the system. In order to do that it implicitly sets the master secret key of the scheme to be $\alpha = xy$, where x, y are set in the assumption. Notice that this way α is properly distributed. Then $\mathcal{B}$ picks the random exponents $\tilde{u}, \tilde{h} \xleftarrow{\$} \mathbb{Z}_p$ and gives to $\mathcal{A}$ the following terms:

$$\begin{array}{ll} g = g & w = g^x \\ u = g^{\tilde{u}} \cdot \prod_{i\in[k]} g^{y/b_i^2} & h = g^{\tilde{h}} \cdot \prod_{i\in[k]} g^{xz/b_i} \cdot \prod_{i\in[k]} \left(g^{y/b_i^2}\right)^{-A_i^*} \\ e(g,g)^\alpha = e(g^x, g^y) & \end{array}$$

Since x is information-theoretically hidden from $\mathcal{A}$, because it is multiplied by y in α, the term w is properly uniformly random in $\mathbb{G}$. The terms u, h are properly distributed due to $\tilde{u}, \tilde{h}$ respectively. Notice that all terms can be calculated by the simulator using suitable terms from the assumption and the challenge set $\mathcal{S}^*$ given by $\mathcal{A}$.

In the KP-ABE proof we see that the "binder term" of the CP-ABE reduction has been contained in the "attribute layer"; namely the g^{xz/b_i} of the h term. Since in KP-ABE the master secret key α is shared in all key generation queries, we don't need any more the extra functionality provided by the powers of a.

Query phases 1 and 2: The simulator has to produce secret keys for policies requested by $\mathcal{A}$, for which the set $\mathcal{S}^*$ is not authorized. In both phases the treatment is the same. We describe here the way $\mathcal{B}$ works in order to create a key for a policy $(M, \rho) \in (\mathbb{Z}_p^{\ell\times n}, \mathcal{F}([\ell] \to \mathbb{Z}_p))$.

Since $\mathcal{S}^*$ is non authorized for (M, ρ), there exists a vector $\vec{w} = (w_1, w_2, \ldots, w_n)^\perp \in \mathbb{Z}_p^n$ such that $w_1 = 1$ and $\langle M_\tau, \vec{w}\rangle = 0$ for all $\tau \in [\ell]$ such that $\rho(\tau) \in \mathcal{S}^*$ (c.f. section 2.2). The simulator calculates $\vec{w}$ using linear algebra. The vector $\vec{y}$ that will be shared is implicitly

$$\vec{y} = xy\vec{w} + (0, \tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n)^\perp$$

where $\tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n \xleftarrow{\$} \mathbb{Z}_p$. This vector is properly distributed because its first component is $xy = \alpha$ and the remaining components are uniformly random in $\mathbb{Z}_p$. Therefore for each row $\tau \in [\ell]$ the share is

$$\lambda_\tau = \langle M_\tau, \vec{y}\rangle = xy\langle M_\tau, \vec{w}\rangle + \langle M_\tau, (0, \tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n)^\perp\rangle = xy\langle M_\tau, \vec{w}\rangle + \tilde{\lambda}_\tau$$

As we mentioned above for each row τ for which $\rho(\tau) \in \mathcal{S}^*$ it is true that $\langle M_\tau, \vec{w}\rangle = 0$. Therefore in this case $\lambda_\tau = \tilde{\lambda}_\tau = \langle M_\tau, (0, \tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n)^\perp\rangle$; hence its value is known to the simulator. In that case it picks $t_\tau \xleftarrow{\$} \mathbb{Z}_p$ and outputs the terms $K_{\tau,0}, K_{\tau,1}, K_{\tau,2}$ as in the KeyGen algorithm.

On the other hand, for each row τ for which $\rho(\tau) \notin \mathcal{S}^*$ it picks $\tilde{t}_\tau \xleftarrow{\$} \mathbb{Z}_p$ and sets implicitly

$$t_\tau = -y\langle M_\tau, \vec{w}\rangle + \sum_{i\in[k]} \frac{xzb_i\langle M_\tau, \vec{w}\rangle}{\rho(\tau) - A_i^*} + \tilde{t}_\tau$$

Since $\rho(\tau) \notin \mathcal{S}^*$ the above fractions are defined and t_τ is properly distributed due to $\tilde{t}_\tau$. The intuition behind this choice is that the y exponent "raises" the power of w to the secret $\alpha = xy$. However, this also results to xyz/b_i exponents from h. Thus, the cancellation is provided by the xzb_i exponents on the y/b_i^2

part. Now the simulator can compute the following terms using the assumption:

$$\begin{aligned}
K_{\tau,0} &= g^{\lambda_\tau} w^{t_\tau} \\
&= g^{xy\langle M_\tau, \vec{w}\rangle + \tilde{\lambda}_\tau} \cdot g^{-xy\langle M_\tau,\vec{w}\rangle + \sum_{i\in[k]} \frac{x^2 z b_i \langle M_\tau, \vec{w}\rangle}{\rho(\tau)-A_i^*}} \cdot w^{\tilde{t}_\tau} \\
&= g^{\tilde{\lambda}_\tau} \cdot \prod_{i\in[n]} \left(g^{x^2 z b_i}\right)^{\langle M_\tau,\vec{w}\rangle/(\rho(\tau)-A_i^*)} \cdot w^{\tilde{t}_\tau}
\end{aligned}$$

$$\begin{aligned}
K_{\tau,1} &= (u^{\rho(\tau)}h)^{-t_\tau} = \\
&= \left(g^{\rho(\tau)\tilde{u}+\tilde{h}} \cdot \prod_{i\in[k]} g^{xz/b_i} \cdot \prod_{i\in[k]} g^{y(\rho(\tau)-A_i^*)/b_i^2}\right)^{y\langle M_\tau,\vec{w}\rangle - \sum_{i\in[k]} \frac{xzb_i\langle M_\tau,\vec{w}\rangle}{\rho(\tau)-A_i^*}} \cdot (u^{\rho(\tau)}h)^{-\tilde{t}_\tau} \\
&= g^{y\langle M_\tau,\vec{w}\rangle(\rho(\tau)\tilde{u}+\tilde{h})} \prod_{i\in[k]} g^{-xzb_i(\rho(\tau)\tilde{u}+\tilde{h})\langle M_\tau,\vec{w}\rangle/(\rho(\tau)-A_i^*)} \\
&\qquad \cdot \prod_{i\in[k]} g^{xyz\langle M_\tau,\vec{w}\rangle/b_i} \prod_{(i,j)\in[k,k]} g^{-(xz)^2 b_j\langle M_\tau,\vec{w}\rangle/b_i(\rho(\tau)-A_j^*)} \\
&\qquad \cdot \prod_{i\in[k]} g^{y^2\langle M_\tau,\vec{w}\rangle(\rho(\tau)-A_i^*)/b_i^2} \prod_{(i,j)\in[k,k]} g^{-xyz\langle M_\tau,\vec{w}\rangle b_j(\rho(\tau)-A_i^*)/b_i^2(\rho(\tau)-A_j^*)} \cdot (u^{\rho(\tau)}h)^{-\tilde{t}_\tau} \\
&= (g^y)^{\langle M_\tau,\vec{w}\rangle(\rho(\tau)\tilde{u}+\tilde{h})} \prod_{i\in[k]} \left(g^{xzb_i}\right)^{-(\rho(\tau)\tilde{u}+\tilde{h})\langle M_\tau,\vec{w}\rangle/(\rho(\tau)-A_i^*)} \\
&\qquad \cdot \prod_{(i,j)\in[k,k]} \left(g^{(xz)^2 b_j/b_i}\right)^{-\langle M_\tau,\vec{w}\rangle/(\rho(\tau)-A_j^*)} \prod_{i\in[k]} \left(g^{y^2/b_i^2}\right)^{\langle M_\tau,\vec{w}\rangle(\rho(\tau)-A_i^*)} \\
&\qquad \cdot \prod_{\substack{(i,j)\in[k,k] \\ i\neq j}} \left(g^{xyzb_j/b_i^2}\right)^{-\langle M_\tau,\vec{w}\rangle(\rho(\tau)-A_i^*)/(\rho(\tau)-A_j^*)} \cdot (u^{\rho(\tau)}h)^{-\tilde{t}_\tau}
\end{aligned}$$

$$\begin{aligned}
K_{\tau,2} &= g^{t_\tau} \\
&= (g^y)^{-\langle M_\tau,\vec{w}\rangle} \cdot \prod_{i\in[k]} \left(g^{xzb_i}\right)^{\langle M_\tau,\vec{w}\rangle/(\rho(\tau)-A_i^*)} \cdot g^{\tilde{t}_\tau}
\end{aligned}$$

Therefore $\mathcal{B}$ can reply to $\mathcal{A}$'s query with the entire secret key $sk = \Big((M,\rho), \{K_{\tau,0}, K_{\tau,1}, K_{\tau,2}\}_{\tau\in[\ell]}\Big)$.

Challenge: The attacker will output a pair of messages (m_0, m_1) of the same length. In this phase the simulator flips a random coin $b \xleftarrow{\$} \{0,1\}$ and sets implicitly $s = z$ from the q-2 assumption. Also, it sets $r_\tau = b_\tau$ for every level $\tau \in [k]$. These parameters are properly distributed since $z, b_1, \ldots, b_q$ are information-theoretically hidden from the attacker's view. Now the simulator can compute the following terms using the assumption:

$$C = m_b \cdot T \qquad C_0 = g^s = g^z$$

$$C_{\tau,1} = g^{r_\tau} = g^{b_\tau}$$

$$\begin{aligned}C_{\tau,2} &= (u^{A^*_\tau}h)^{r_\tau}\cdot w^{-s}\\ &= g^{b_\tau(\tilde{u}A^*_\tau+\tilde{h})}\cdot\prod_{i\in[k]} g^{xzb_\tau/b_i}\prod_{i\in[k]} g^{yb_\tau(A^*_k-A^*_i)/b_i^2}\cdot g^{-xz}\\ &= \left(g^{b_\tau}\right)^{\tilde{u}A^*_\tau+\tilde{h}}\cdot\prod_{\substack{i\in[k]\\ i\neq\tau}} g^{xzb_\tau/b_i}\prod_{\substack{i\in[k]\\ i\neq\tau}}\left(g^{yb_\tau/b_i^2}\right)^{A^*_\tau-A^*_i}\end{aligned}$$

As one can see, the choice of $r_\tau = b_\tau$ "raises" one of the xz/b_i components to xz and achieves the cancellation with w^{-s}. The simulator hands over the ciphertext $ct = \left(\mathcal{S}^*, C, C_0, \{C_{\tau,1}, C_{\tau,2}\}_{\tau\in[k]}\right)$ to the attacker $\mathcal{A}$.

Guess: After the query phase 2, where the simulator creates the secret keys as described above, the attacker outputs a guess b' for the challenge bit. If $b' = b$ the simulator outputs 0, i.e. it claims that the challenge term is $T = e(g,g)^{xyz}$. Otherwise, it outputs 1.

If $T = e(g,g)^{xyz}$ then $\mathcal{A}$ played the proper security game, because $C = m_b\cdot T = m_b\cdot e(g,g)^{\alpha s}$. On the other hand, if T is a random term of $\mathbb{G}_T$ then all information about the message m_b is lost in the challenge ciphertext. Therefore the advantage of $\mathcal{A}$ is exactly 0. As a result if $\mathcal{A}$ breaks the proper security game with a non negligible advantage, then $\mathcal{B}$ has a non negligible advantage in breaking the q-2 assumption. ■

D Generic Security of the Assumptions

In this section we consider the security of our assumptions in the generic group model introduced by Shoup [39]. In this model the attacker does not receive the actual representations of group elements in $\mathbb{G}$ or $\mathbb{G}_T$, but handles picked from a sufficiently large handle space. Whenever a new group element has to be given to the attacker, he receives a uniformly random handle from the handle space; not picked before. From now on this handle is "fixed" to this specific group element. The attacker is allowed to query for operations on the handles he has already received. Then the challenger executes the operation on the underlying group elements and returns either a freshly picked handle, if the result is new, or an existing handle. The only operations available on group elements to the attacker are multiplications in $\mathbb{G}$ or $\mathbb{G}_T$, pairings in $\mathbb{G}$ and the equality checking of two group elements in $\mathbb{G}$ or $\mathbb{G}_T$ by checking the equality of handles. The main goal of the generic group model proofs is to provide an indication of the absence of "security holes" which are independent from the specific group representations.

The following theorem will provide an easier way to argue about the security of our assumptions in the generic group model.

Definition D.1 ($\mathbb{G}_T$-monomial assumption)**.** *A $\mathbb{G}_T$-monomial assumption is parameterized by a security parameter $\lambda \in \mathbb{N}$. It refers to a prime order bilinear group $D = (p, \mathbb{G}, \mathbb{G}_T, e)$ with $p = \Theta(2^\lambda)$, a matrix $A \in \mathbb{Z}^{L\times K}$ and two target vectors $\tilde{A}^0, \tilde{A}^1 \in \mathbb{Z}^{1\times K}$.*

We require that $\tilde{A}^0 \neq \tilde{A}^1$ and that the natural numbers K, L and the the absolute values of the integers in A, $\tilde{A}^0$ and $\tilde{A}^1$ are all polynomially bounded in λ.

The assumption is defined via a game between a challenger and an adversary. Initially, the challenger picks K independent and uniformly random *variables $X_1, X_2, \ldots, X_K \xleftarrow{\$} \mathbb{Z}_p$. Then it constructs the following monomials in these variables:*

$$Y_i = \prod_{j\in[K]} X_j^{A_{i,j}} \text{ for all } i \in [L] \quad , \quad Z_0 = \prod_{j\in[K]} X_j^{\tilde{A}^0_j} \text{ and } Z_1 = \prod_{j\in[K]} X_j^{\tilde{A}^1_j}$$

Finally, the challenger picks $g \xleftarrow{\$} \mathbb{G}$ and $b \xleftarrow{\$} \{0,1\}$. He sends to the adversary the description of the group $D = (p, \mathbb{G}, \mathbb{G}_T, e)$, the matrix A, the vectors $\tilde{A}^0, \tilde{A}^1$, the terms $\left\{g^{Y_i}\right\}_{i\in[L]}$, and the challenge term $e(g,g)^{Z_b}$. The assumption claims that no PPT adversary has a non negligible advantage in guessing the bit b.

We will prove the following theorem that refers to the generic security of a $\mathbb{G}_T$-monomial assumption:

Theorem D.2. *The above assumption is secure in the generic group model if and only if for all $i, j \in [L]$ it is true that $\tilde{A}^0 \neq A^i + A^j$ and $\tilde{A}^1 \neq A^i + A^j$, where A^i is the i-th row of A.*

For the proof of the theorem D.2 we will use the following lemma:

Lemma D.3. *Consider any linear combination of the form*

$$T(X_1, X_2, \ldots, X_K) = \tilde{c}_0 Z_0 + \tilde{c}_1 Z_1 + \sum_{(i,j)\in[L,L]} c_{i,j} Y_i \cdot Y_j$$

with $\tilde{c}_0, \tilde{c}_1, c_{i,j}$ constants in $\mathbb{Z}_p$ and T is not identically zero as a rational function in variables X_1, X_2, ..., X_K.

Then the probability that $T(X_1, X_2, \ldots, X_K) = 0 \pmod{p}$ is negligible in λ.

Proof of lemma D.3. The proof is an immediate consequence of the Schwartz-Zippel lemma and the fact that the total degree of each monomial is polynomially bounded in λ. More specifically, consider the polynomial

$$T'(X_1, X_2, \ldots, X_K) = T(X_1, X_2, \ldots, X_K) \cdot C(X_1, X_2, \ldots, X_K)$$

where $C(X_1, X_2, \ldots, X_K) = \prod X_i^{d_i}$ with d_i being the absolute value of the minimum negative exponent of X_i in the monomials Y_iY_j for any i, j and the Z_0, Z_1, or 0 if X_i has only positive exponents. As a result T' is a polynomial in variables X_1, X_2, ..., X_K.

Since the absolute values of the elements of A, $\tilde{A}^0$, and $\tilde{A}^1$ are all polynomially bounded in λ, the total degree of the polynomial T' is also polynomially-bounded in λ. Since T, and therefore T', is not identically zero we can apply the Schwartz-Zippel lemma: the probability that T' becomes zero is at most $\mathcal{O}(\lambda^c)/p = \mathcal{O}(\lambda^c)/\Theta(2^\lambda) = \mathsf{negl}(\lambda)$. This is equal to the probability that T is zero or undefined (when some X_i with $d_i > 0$ is instantiated to zero). Therefore, the probability that T is zero is at most negligible in λ. ■

Proof of theorem D.2. We will prove the forward direction first. Namely, suppose that there exist i, j and $b' \in \{0, 1\}$ such that $\tilde{A}^{b'} = A^i + A^j$. W.l.o.g. we assume $b' = 0$. Since according to the definition of the assumption $\tilde{A}^0 \neq \tilde{A}^1$, we conclude that $\tilde{A}^1 \neq A^i + A^j$.

The adversary can in polynomial time find these i, j, because L is polynomial, and request the handle of the term $e(g^{Y_i}, g^{Y_j})$. If this handle is equal to the handle of $e(g, g)^{Z_b}$ of the challenge term, it outputs 0. Otherwise it outputs 1.

Therefore, if b is indeed 0 the adversary is successful with certainty. If it is the case that $b = 1$, the adversary makes a wrong guess only when the handle of $e(g, g)^{Z_1}$ happens to be the same as the handle of $e(g^{Y_i}, g^{Y_j})$. According to the generic group model this is equivalent to $Z_1 = Y_i \cdot Y_j$ (after the instantiations). Since $\tilde{A}^1 \neq A^i + A^j$ we get that the expression $Z_1 - Y_i \cdot Y_j$ is not identically zero. Therefore, according to lemma D.3 we conclude that the error probability of the adversary when $b = 1$ is $\mathsf{negl}(\lambda)$. As a result the advantage of the adversary is non negligible and the assumption not secure in the generic group model.

For the backward direction we assume that there exists a PPT adversary that breaks the assumption in the generic group model game. First, we define a new security game where the random variables $X_1, X_2, \ldots, X_K$ are never instantiated and the handles returned to the adversary are the same only when the rational functions of the X_i's in the exponents of group elements are formally equal. This game differs from the real generic group model game only when two different linear combinations of monomials are instantiated to the same value. Since the number of queries by the adversary is polynomial and because of lemma D.3, the probability of this event is negligible. Therefore, the adversary has a non negligible advantage in the modified game.

Since the only decision query he can ask is to compare the handles of two terms, he can construct two terms $T_1, T_2 \in \mathbb{G}_T$ such that $T_1 = T_2$ (in terms of formal equality of the exponents) for one value of b but not for the other[6]. According to the allowable operations and the terms given to adversary, T_1 and T_2 should be of the form $e(g, g)^S$ where S is a linear combination of the set of monomials $\{Z_b\} \cup \{Y_i \cdot Y_j\}_{i,j\in[L]}$.

[6]A decision in $\mathbb{G}$ can be expressed in terms of $\mathbb{G}_T$ by pairing with the same element of $\mathbb{G}$

Type	Given Terms	Conditions	a	s	b_1	b_2	$\ldots$	b_q
1	g		0	0	0	0	$\ldots$	0
2	g^s		0	1	0	0	$\ldots$	0
3	g^{a^i}	$\forall i \in [q]$	i	0	0	0	$\ldots$	0
4	g^{b_j}	$\forall j \in [q]$	0	0	$[j : 1]$			
5	g^{sb_j}	$\forall j \in [q]$	0	1	$[j : 1]$			
6	$g^{a^i b_j}$	$\forall (i,j) \in [q,q]$	i	0	$[j : 1]$			
7	g^{a^i / b_j^2}	$\forall (i,j) \in [q,q]$	i	0	$[j : (-2)]$			
8	g^{a^i / b_j}	$\forall (i,j) \in [2q,q]$ with $i \neq q+1$	i	0	$[j : (-1)]$			
9	$g^{a^i b_j / b_{j'}^2}$	$\forall (i,j,j') \in [2q,q,q]$ with $j \neq j'$	i	0	$[j : 1, j' : (-2)]$			
10	$g^{s a^i b_j / b_{j'}}$	$\forall (i,j,j') \in [q,q,q]$ with $j \neq j'$	i	1	$[j : 1, j' : (-1)]$			
11	$g^{s a^i b_j / b_{j'}^2}$	$\forall (i,j,j') \in [q,q,q]$ with $j \neq j'$	i	1	$[j : 1, j' : (-2)]$			
$\tilde{A}^0$	$e(g,g)^{s a^{q+1}}$		$q+1$	1	0	0	$\ldots$	0

Table 4: Compact form of matrix A and target vector $\tilde{A}^0$ for the q-1 assumption.

W.l.o.g. suppose that T_1 is equal to T_2 when $b = 0$ and different otherwise. Then if $T_1 = e(g,g)^{S_1}$ and $T_2 = e(g,g)^{S_2}$, we get that $T_1 = T_2 \implies S_1 = S_2 \implies Z_0 = S^*$, where S^* is a linear combination of only the monomials $\{Y_i \cdot Y_j\}_{i,j \in [L]}$. The coefficient of Z_0 has to be non-zero because otherwise the value of b would be information-theoretically hidden and the advantage of the adversary would be zero in this game. Since the $Z_0 = S^*$ is a formal equation and Z_0 is a monomial the only way this is possible is to have $Z_0 = Y_i \cdot Y_j$ for some i, j. Therefore, $\tilde{A}^0 = A^i + A^j$. ■

The following corollary refers to a $\mathbb{G}_T$-monomial assumption where the second challenge term is uniformly random from $\mathbb{G}_T$. The proof from theorem D.2 is trivial and is omitted.

Corollary D.4. *If $\tilde{A}^1 = (0, 0, \ldots, 0, 1) \in \mathbb{Z}^{1 \times K}$ and $\langle \tilde{A}^1, A^i \rangle = 0$ for all $i \in [L]$, the corresponding $\mathbb{G}_T$-monomial assumption is secure in the generic group model if and only if for all $i, j \in [L]$ it is true that $\tilde{A}^0 \neq A^i + A^j$.*

Using the above corollary we show that our assumptions are secure in the generic group model in lemmata D.5 and D.6.

Lemma D.5. *The q-1 assumption is secure in the generic group model.*

Proof. First, notice that this is indeed a $\mathbb{G}_T$-monomial assumption with random variables a, s, b_1, b_2, $\ldots$, b_q instead of X_1, X_2, $\ldots$, X_{K-1}. X_K is the uniformly random exponent of the second challenge term; not present in any of the remaining terms. Thus corollary D.4 applies.

In table 4 we denote by $[i : x]$ and $[i : x, i' : y]$ the row vectors in $\mathbb{Z}^{1 \times q}$ with all components equal to 0, except the i-th component for the first vector and the i, i'-th components for the second. The non zero elements are x for the first vector and x, y for the i, i'-th positions, respectively, of the second vector. The table shows a compact form of the matrix A where rows of similar type are shown in one line.

In order to prove the lemma we have to show that by adding any two rows of matrix A we can not get the row vector $\tilde{A}^0 = (q+1, 1, 0, 0, \ldots, 0)$. By inspecting table 4 we can easily see that we have to check only the rows of types 2, 5, 10, and 11, which have 1 in the s column.

The only rows that can be added to row 2 and give all zero's in the b_i columns are row 1 or rows of type 3. But in both of them we can not get the $q+1$ component in the a column. Rows of type 5 can be

Type	Given Terms	Conditions	x	y	z	b_1	b_2	$\dots$	b_q
1	g		0	0	0	0	0	$\dots$	0
2	g^x		1	0	0	0	0	$\dots$	0
3	g^y		0	1	0	0	0	$\dots$	0
4	g^z		0	0	1	0	0	$\dots$	0
5	$g^{(xz)^2}$		2	0	2	0	0	$\dots$	0
6	g^{b_i}	$\forall i \in [q]$	0	0	0	$[i : 1]$			
7	g^{xzb_i}	$\forall i \in [q]$	1	0	1	$[i : 1]$			
8	g^{xz/b_i}	$\forall i \in [q]$	1	0	1	$[i : (-1)]$			
9	$g^{x^2zb_i}$	$\forall i \in [q]$	2	0	1	$[i : 1]$			
10	g^{y/b_i^2}	$\forall i \in [q]$	0	1	0	$[i : (-2)]$			
11	g^{y^2/b_i^2}	$\forall i \in [q]$	0	2	0	$[i : (-2)]$			
12	g^{xzb_i/b_j}	$\forall (i,j) \in [q,q]$ with $i \neq j$	1	0	1	$[i : 1, j : (-1)]$			
13	g^{yb_i/b_j^2}	$\forall (i,j) \in [q,q]$ with $i \neq j$	0	1	0	$[i : 1, j : (-2)]$			
14	g^{xyzb_i/b_j^2}	$\forall (i,j) \in [q,q]$ with $i \neq j$	1	1	1	$[i : 1, j : (-2)]$			
15	$g^{(xz)^2b_i/b_j}$	$\forall (i,j) \in [q,q]$ with $i \neq j$	2	0	2	$[i : 1, j : (-1)]$			
$\tilde{A}^0$	$e(g,g)^{xyz}$		1	1	1	0	0	$\dots$	0

Table 5: Compact form of matrix A and target vector $\tilde{A}^0$ for the q-2 assumption.

added only to rows of type 8 and give only zeros in the b_i columns. But the term with $i = q+1$ is excluded from rows of type 8; therefore the target vector can not be obtained. Finally, rows of type 10 or 11 can not be added to one of the rows 1, 3, 4, 6, 7, 8, and 9 without having at least one non zero element in the b_i columns. That is because none of these rows have vectors of the form $[j : (-1), j' : 1]$ or $[j : (-1), j' : 2]$, which are needed to cancel the b_i components.

Therefore, according to corollary D.4 the q-1 assumption is secure in the generic group model. ■

Lemma D.6. *The q-2 assumption is secure in the generic group model.*

Proof. q-2 is a $\mathbb{G}_T$-monomial assumption with random variables x, y, z, b_1, b_2, …, b_q instead of X_1, X_2, …, X_{K-1}. The matrix A and the target vector $\tilde{A}^0$ for this assumption are shown in table 5.

In order to prove the lemma we have to show that by adding any two rows of matrix A we can not get the row vector $\tilde{A}^0 = (1,1,1,0,0,\dots,0)$. We will mainly focus on the first three columns of matrix A, i.e. the x, y, z columns. First, we observe that the rows of types 5, 9, 11, and 15, can not be used since they have at least one 2 in the first three columns and all other rows are positive in these columns. The rows 2 and 4 have "100" and "001" in the first three columns, respectively. Since there are no rows with "011" or "110", they can not be used to give the required "111". Row 1 or a row of type 6 can only be combined to a row of type 14, and vice versa, because the former have "000" and the later "111". But since a row of type 14 has $[i : 1, j : (-2)]$ in the b_i columns, it can not be used to give all zeros in them. Finally, rows of type 7, 8, or 12, that have "101" in the first three columns, might be possibly combined to row 3 or rows of type 10, or 13, and vice versa. However, this still won't give the target vector, because none of the partial vectors $[i : 1]$,

$[i:(-1)]$, and $[i:1, j:(-1)]$ can be added to any of the vectors $(0,0,\dots,0)$, $[i:(-2)]$, and $[i:1, j:(-2)]$, and give the all zero vector in the b_i columns.

Therefore, according to corollary D.4 the q-2 assumption is secure in the generic group model. ■