Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization

Abstract

We present a new methodology for realizing Ciphertext-Policy Attribute Encryption (CP-ABE) under concrete and noninteractive cryptographic assumptions in the standard model. Our solutions allow any encryptor to specify access control in terms of any access formula over the attributes in the system. In our system, ciphertext size, encryption, and decryption time scales linearly with the complexity of the access formula. The only previous work to achieve these parameters was limited to a proof in the generic group model.

Our main system is proven selectively secure under a assumption that we call the decisional Parallel Bilinear Diffie-Hellman Exponent (PBDHE) assumption which can be viewed as a generalization of the BDHE assumption. In addition, we provide two variants of our construction that provide performance tradeoffs to achieve provable security respectively under the (weaker) decisional Bilinear-Diffie-Hellman Exponent and decisional Bilinear Diffie-Hellman assumptions.

1 Introduction

Public-Key encryption is a powerful mechanism for protecting the confidentiality of stored and transmitted information. Traditionally, encryption is viewed as a method for a user to share data to a targeted user or device. While this is useful for applications where the data provider knows specifically which user he wants to share with, in many applications the provider will want to share data according to some policy based on the receiving user's credentials.

Sahai and Waters [31] presented a new vision for encryption where the data provider can express how he wants to share data in the encryption algorithm itself. The data provider will provide a predicate $f(\cdot)$ describing how he wants to share the data and a user will be ascribed a secret key associated with their credentials X; the user with credentials X can decrypt a ciphertext encrypted with predicate f if $f(X) = 1$. Sahai and Waters [31] presented a particular formulation of this problem that they called Attribute-Based Encryption (ABE), in which a user's credentials is represented by a set of string called "attributes" and the predicate is represented by a formula over these attributes. Several techniques used by SW were inspired by prior work on Identity-Based Encryption [32, 11, 22, 16, 8]. One drawback of the Sahai-Waters approach is that their initial construction was limited to handling formulas consisting of one threshold gate.

In subsequent work, Goyal, Pandey, Sahai, and Waters [26] further clarified the concept of Attribute-Based Encryption. In particular, they proposed two complementary forms of ABE. In the first, Key-Policy ABE, attributes are used to annotate the ciphertexts and formulas over these attributes are ascribed to users' secret keys. The second type, Ciphertext-Policy ABE, is complementary in that attributes are used to describe the user's credentials and the formulas over these credentials are attached to the ciphertext by the encrypting party. In addition, Goyal et al. [26]

provided a construction for Key-Policy ABE that was very expressive in that in allowed keys to be expressed by *any* monotonic formula over encrypted data. The system was proved selectively secure under the Bilinear Diffie-Hellman assumption. However, they left creating expressive Ciphertext Policy ABE schemes as an open problem.

The first work to explicitly address the problem of Ciphertext-Policy Attribute-Based Encryption was by Bethencourt, Sahai, and Waters [7]. They described an efficient system that was expressive in that it allowed an encryptor to express an access predicate f in terms of any monotonic formula over attributes. Their system achieved analogous expressiveness and efficiency to the Goyal et al. construction, but in the Ciphertext-Policy ABE setting. While the BSW construction is very expressive, the proof model used was less than ideal — the authors only showed the scheme secure in the generic group model, an artificial model which assumes the attacker needs to access an oracle in order to perform any group operations.[1]

Ciphertext Policy ABE in the Standard Model The lack of satisfaction from with generic group model proofs is motivated the problem of finding an expressive CP-ABE system under a more solid model. There have been multiple approaches in this direction.

First, we can view the Sahai-Waters[31] construction most "naturally" as Key-Policy ABE for a threshold gate. In their work, Sahai and Waters describe how to realize Ciphertext-Policy ABE for threshold gates by "grafting" so called "dummy attributes" over their basic system. Essentially, they transformed a KP-ABE system into a CP-ABE one with the expressiveness of a single threshold gate. [2] Cheung and Newport[21] provide a direct construction for constructing a policy with a single AND gate under the Bilinear Diffie-Hellman assumption. Their approach has the drawbacks that it only allows a fixed number of system attributes and is limited to an AND gate (does not enable thresholds). In retrospect these two limitations actually make it less expressive than the SW transformation, although this wasn't necessarily immediately apparent.

Most recently, Goyal, Jain, Pandey, and Sahai [25] generalized the transformational approach to show how to transform a KP-ABE system into a CP-ABE one using what they call a "universal access tree". In particular, they provided a mapping onto a "universal" (or complete) access tree of up to depth d formulas consisting of threshold gates of input size m, where m and d are chosen by the setup algorithm. They applied a similar "dummy attribute" approach.

In order to accommodate a general access formula of size n, their scheme first translates this into a balanced formula. Under standard techniques a formula of size n can be "balanced" such that any formula (tree) of size n can be covered by a complete tree of size approximately $O(n^{3.42})$. Their work was the first feasibility result for expressive CP-ABE under a non-interactive assumption. Unfortunately, the parameters of ciphertext and private key sizes add encryption and decryption complexity blow up (in the worst case) by an $n^{3.42}$ factor limiting its usefulness in practice. For instance, in a system with U attributes defined and n nodes the ciphertext overhead will be approximately a factor of $U \cdot n^{2.42}$ greater than that of the BSW system. To give a concrete example, for the modest parameters of universe size $U = 100$ attributes and a formula of 20 nodes the blowup factor relative to BSW is approximately $140,000$.

[1]Alternatively, we could derive a concrete, but interactive and complicated assumption directly from the scheme itself and argue that the scheme is secure under this assumption. However, this view is also not very satisfactory.

[2]The Sahai-Waters construction was given prior to the Key-Policy and Ciphertext-Policy distinction; our interpretation is a retrospective one.

Our Contribution We present a new methodology for realizing Ciphertext-Policy ABE systems from a general set of access structures in the *standard model* under concrete and non-interactive assumptions. Both the ciphertext overhead and encryption time scale with $O(n)$ where n is the size of the formula. In addition, decryption time scales with the number of nodes.

Our system allows an encryption algorithm to specify an access formula in terms of any access formula. In fact our techniques are slightly more general. We express access control by a Linear Secret Sharing Scheme (LSSS) matrix M over the attributes in the system. Previously used structures such as formulas (equivalently tree structures) can be expressed succinctly [6] in terms of a LSSS. *We do not loose any efficiency by using the more general LSSS representation as opposed to the previously used tree access structure descriptions.* Thus, we achieve the same performance and functionality as the Bethencourt, Sahai, and Waters construction, but under the standard model.

In addition, we provide two other variants of our construction that tradeoff some performance parameters for provable security under the respective weaker assumptions of decisional-Bilinear Diffie-Hellman Exponent (d-BDHE) and decisional-Bilinear Diffie-Hellman assumptions. In Table 1 we summarize the comparisons between our schemes and the GJPS and BSW CP-ABE systems in terms of ciphertext and key sizes and encryption and decryption times. Taken all together our main scheme realizes the same efficiency parameters as the BSW encryption scheme, but under a concrete security assumption. At the same time, our d-BDH variant is proved under the same assumption as the GJPS system and achieves significantly better performance.

Our Techniques Our techniques provide a framework for *directly* realizing provably secure CP-ABE systems. In our systems, the ciphertext distributes shares of a secret encryption exponent s across different attributes according to the access control LSSS matrix M.

A user's private key is associated with a set S of attributes and he will be able to decrypt a ciphertext iff his attributes "satisfy" the access matrix associated with the ciphertext. As in previous ABE systems, the primary challenge is to prevent users from realizing collusion attacks. Our main tool to prevent this is to randomize each key with an freshly chosen exponent t. During decryption, each share will be multiplied by a factor t in the exponent. Intuitively, this factor should "bind" the components of one user's key together so that they cannot be combined with another user's key components. During decryption, the different shares (in the exponent) that the algorithm combines are multiplied by a factor of t. Ultimately, these randomized shares are only useful to that one particular key.

Our construction structure and high level intuition for security is similar to the BSW construction. The main novelty in our paper is provide a method for proving security of such a construction. The primary challenge one comes across is (in the selective model) how to create a reduction that embeds a complex access structure in a short number of parameters. All prior ABE schemes follow a "partitioning" strategy for proving security where the reduction algorithm sets up the public parameters such that it knows all the private keys that it needs to give out, yet it cannot give out private keys that can trivially decrypt the challenge ciphertext. In prior KP-ABE schemes the challenge ciphertext was associated with a set S^* of attributes. This structure could fairly easily be embedded in a reduction as the public parameter for each attribute was simply treated differently depending whether or not it was in S^*. In CP-ABE, the situation is much more complicated as ciphertexts are associated with a potentially large access structure M^* that includes attributes multiple times. In general, the size of M^* is much larger than the size of the public parameters. [3]

[3]Here we roughly mean size to be number of rows in the LSSS system or nodes in an access tree.

System	Ciphetext Size	Private Key Size	Enc. Time	Dec. Time	Assumption
BSW[7]	$\mathcal{O}(n)$	$\mathcal{O}(A)$	$\mathcal{O}(n)$	$\mathcal{O}(T)$	Generic Group
GJPS[25]	$\mathcal{O}(U \cdot n_{\max}^{3.42})$	$\mathcal{O}(A \cdot n_{\max}^{3.42})$	$\mathcal{O}(U \cdot n_{\max}^{3.42})$	$\mathcal{O}(U \cdot n_{\max}^{3.42})$	d-BDH
Main Construction	$\mathcal{O}(n)$	$\mathcal{O}(A)$	$\mathcal{O}(n)$	$\mathcal{O}(T)$	d-Parallel BDHE
Appendix A	$\mathcal{O}(n)$	$\mathcal{O}(k_{\max} \cdot A)$	$\mathcal{O}(n)$	$\mathcal{O}(T)$	d-BDHE
Appendix B	$\mathcal{O}(n^2)$	$\mathcal{O}(k_{\max} \cdot A + n_{\max})$	$\mathcal{O}(n^2)$	$\mathcal{O}(n \cdot T)$	d-BDH

Table 1: Comparison of CP-ABE systems in terms of ciphertext size, private key size, encryption and decryption times and assumptions. We let n be the size of an access formula , A be the number of attributes in a user's key, and T be (minimum needed) number of nodes satisfied of a formula by a user's attributes. For our d-BDHE construction of Appendix A the system defines a parameter $k_{\max}$, which is the maximum number of times a single attribute will appear in a particular formula. In the GJPS construction and our d-BDH one of Appendix B the systems define $n_{\max}$ as a bound on the size any formula. The ciphertext and private key sizes are given in terms of the number of group elements, encryption time in terms of number of exponentiations, and decryption in terms of number of pairing operations.

Consequently, there is not a simple "on or off" method of programming this into the parameters. Arguably, it is this challenge that lead the BSW paper to apply the generic group heuristic and GJPS paper to translate the problem back to KP-ABE.

In this paper, we create a method for directly embedding any LSSS structure M^* into the public parameters in our reduction. In the proofs of our system a simulator can "program" the LSSS matrix M^* of the challenge ciphertext (in the selective model of security). Consider a LSSS matrix M^* of size $l^* \times n^*$. For each row i of M^* the simulator needs to program in ℓ pieces of information $(M^*_{i,1}, \ldots, M^*_{i,\ell})$ into the parameters related to the attribute assigned to that row. In our main system we program in M^* using the d-Parallel BDHE assumption; however, in Appendices A and B we show variations of our construction that are provably secure using similar ideas, but under weaker assumptions.

Our methodology of creating a system and proof that directly addresses CP-ABE stands in contrast to the approach of GJPS which essentially maps CP-ABE requirements onto a KP-ABE scheme.

1.1 Related Work

Some of the roots of ABE can be traced back to Identity-Based Encryption [32, 11, 22, 16, 8, 36, 23, 12] (IBE). One can view IBE as a very special case of ABE.

Different authors [34, 29, 4, 15, 3, 5] have considered similar problems without considering collusion resistance. In these works a data provider specifies an access formula by such that a group of users can decrypt if the *union* of their credentials satisfies the formula. By only requiring the union of the credentials one does not worry about collusion attacks. In these schemes a setup authority simply assigns a separate public key to each credential and gives the corresponding secret key to each user that possesses the credential. Encryption is done by splitting secrets and then encrypting each share to the appropriate public key. Some of these schemes were inspired by earlier work [20, 19].

Since the introduction of Attribute-Based Encryption by Sahai and Waters [31], there have been several papers [26, 7, 18, 30, 25] that have proposed different varieties of ABE. Most of them have been for monotonic access structures over attributes; one exception is the work of Ostrovsky, Sahai, and Waters [30] that showed how to realize negation by integrating revocation schemes into the GPSW ABE cryptosystem.

Most work on ABE is focused on complex access controls for hiding an encrypted payload of data. A related line of work called predicate encryption or searching on encrypted data attempts to evaluate predicates over the encrypted data itself [35, 10, 1, 14, 13, 33, 28]. These systems have the advantages of hiding the associated access structures themselves and thus providing a level of "anonymity". The concept of predict encryption is more general than the one we consider. However, the predicate encryption systems realized thus far tend to be much less expressive than access control systems that leave the access structures in the clear.

Other examples of encryption systems with more "structure" added include Hierarchical Identity-Based Encryption [27, 24] and Wildcard IBE [2].

2 Background

We first give formal definitions for access structures and relevant background on Linear Secret Sharing Schemes (LSSS). Then we give the security definitions of ciphertext policy attribute based encryption (CP-ABE). Finally, we give background information on bilinear maps.

2.1 Access Structures

Definition 1 (Access Structure [6]). *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$:* if *$B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An* access structure *(respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the* authorized sets*, and the sets not in $\mathbb{A}$ are called the* unauthorized sets.

In our context, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. We restrict our attention to monotone access structures. However, it is also possible to (inefficiently) realize general access structures using our techniques by having the not of an attribute as a separate attribute altogether. Thus, the number of attributes in the system will be doubled. From now on, unless stated otherwise, by an access structure we mean a monotone access structure.

2.2 Linear Secret Sharing Schemes

We will make essential use of linear secret-sharing schemes. We adapt our definitions from those given in [6]:

Definition 2 (Linear Secret-Sharing Schemes (LSSS)). *A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called* linear *(over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*

2. *There exists a matrix M called the share-generating matrix for Π. The matrix M has ℓ rows and n columns. For all $i = 1, \ldots, \ell$, the i'th row of M we let the function ρ defined the party*

labeling row i as $\rho(i)$. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.

It is shown in [6] that every linear secret sharing-scheme according to the above definition also enjoys the *linear reconstruction* property, defined as follows: Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$.

Furthermore, it is shown in [6] that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

Using Access Trees Prior works on ABE (e.g., [26]) typically described access formulas in terms of binary trees. Using standard techniques one can convert any monotonic boolean formula into an LSSS representation. An access tree of ℓ nodes will result in an LSSS matrix of ℓ rows.same

2.3 Ciphertext-Policy ABE

An ciphertext-policy attribute based encryption scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

Setup(λ, U). The setup algorithm takes security parameter and attribute universe description as input. It outputs the public parameters PK and a master key MK.

Encrypt($\mathrm{PK}, M, \mathbb{A}$). The encryption algorithm takes as input the public parameters PK, a message M, and an access structure $\mathbb{A}$ over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains $\mathbb{A}$.

Key Generation(MK, S). The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.

Decrypt($\mathrm{PK}, \mathrm{CT}, \mathrm{SK}$). The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy $\mathbb{A}$, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure $\mathbb{A}$ then the algorithm will decrypt the ciphertext and return a message M.

We now describe a security model for ciphertext-policy ABE schemes. Like identity-based encryption schemes [32, 11, 22] the security model allows the adversary to query for any private keys that cannot be used to decrypt the challenge ciphertext. In CP-ABE the ciphertexts are identified with access structures and the private keys with attributes. It follows that in our security definition the adversary will choose to be challenged on an encryption to an access structure $\mathbb{A}^*$ and can ask for any private key S such that S does not satisfy $\mathbb{S}^*$. We now give the formal security game.

Security Model for CP-ABE

Setup. The challenger runs the Setup algorithm and gives the public parameters, PK to the adversary.

Phase 1. The adversary makes repeated private keys corresponding to sets of attributes $S_1, \ldots, S_{q_1}$.

Challenge. The adversary submits two equal length messages M_0 and M_1. In addition the adversary gives a challenge access structure $\mathbb{A}^*$ such that none of the sets $S_1, \ldots, S_{q_1}$ from Phase 1 satisfy the access structure. The challenger flips a random coin b, and encrypts M_b under $\mathbb{A}^*$. The ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restriction that none of sets of attributes $S_{q_1+1}, \ldots, S_q$ satisfy the access structure corresponding to the challenge.

Guess. The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$. We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 3. *An ciphertext-policy attribute-based encryption scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game.*

We say that a system is *selectively* secure if we add an Init stage before setup where the adversary commits to the challenge access structure $\mathbb{A}^*$.

2.4 Bilinear Maps

We present a few facts related to groups with efficiently computable bilinear maps and then give our number theoretic assumptions.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G}\times\mathbb{G} \rightarrow \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.4.1 Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption

We define the decisional q-parallel Bilinear Diffie-Hellman Exponent problem as follows. Choose a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a, s \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. If an adversary is given $\vec{y}$=

$$\begin{array}{ll} & g, g^s, g^a, \ldots, g^{(a^q)}, \ , g^{(a^{q+2})}, \ldots, g^{(a^{2q})} \\ \forall_{1 \le j \le q} & g^{s \cdot b_j}, \quad g^{a/b_j}, \ldots, g^{(a^q/b_j)}, \ , g^{(a^{q+2}/b_j)}, \ldots, g^{(a^{2q}/b_j)} \\ \forall_{1 \le j,k \le q, k \neq j} & g^{a \cdot s \cdot b_k/b_j}, \ldots, g^{(a^q \cdot s \cdot b_k/b_j)} \end{array}$$

it must remain hard to distinguish $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving decisional q-parallel BDHE in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\big(\vec{y}, T = e(g,g)^{a^{q+1}s}\big) = 0\right] - \Pr\left[\mathcal{B}\big(\vec{y}, T = R\big) = 0\right] \right| \geq \epsilon$$

Definition 2.1. *We say that the (decision) q parallel-BDHE assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional q-parallel BDHE problem.*

3 Our Construction

We now give our main construction that both realizes expressive functionality and is efficient and is provably secure under a concrete,non-interactive assumption.

In our construction the encryption algorithm will take as input a LSSS access matrix M and distribute a random exponent $s \in \mathbb{Z}_p$ according to M. Private keys are randomized to avoid collusion attack.

Setup(U) The setup algorithm takes as input the number of attributes in the system. It then chooses a group $\mathbb{G}$ of prime order p, a generator g and U random group element $h_1, \ldots, h_U \in \mathbb{G}$ that are associated with the U attributes in the system. In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$.

The public key is published as

$$\mathrm{PK} = \quad g,\ e(g,g)^{\alpha},\ g^a,\ h_1, \ldots, h_U.$$

The authority sets $\mathrm{MSK} = g^{\alpha}$ as the master secret key.

Encrypt($\mathrm{PK}, (M,\rho), \mathcal{M}$) The encryption algorithm takes as input the public parameters PK and a message $\mathcal{M}$ to encrypt. In addition, it takes as input an LSSS access structure (M,ρ). The function ρ associates rows of M to attributes. In this construction we limit ρ to be an injective function, that is an attribute is associated with at most one row of M.

Let M be an $\ell \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For $i = 1$ to ℓ, it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. In addition, the algorithm chooses random $r_1, \ldots, r_\ell \in \mathbb{Z}_p$.

The ciphertext is published as CT =

$$C = \mathcal{M}e(g,g)^{\alpha s},\ C' = g^s, (C_1 = g^{a\lambda_1} h_{\rho(1)}^{-r_1},\ D_1 = g^{r_1}), \ldots, (C_n = g^{a\lambda_n} h_{\rho(n)}^{-r_n},\ D_n = g^{r_n})$$

along with a description of (M,ρ).

KeyGen(MSK, S) The key generation algorithm takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t \in \mathbb{Z}_p$. It creates the private key as

$$K = g^{\alpha} g^{at} \quad L = g^t \quad \forall x \in S\ K_x = h_x^t.$$

Decrypt(CT,SK) The decryption algorithm takes as input a ciphertext CT for access structure (M, ρ) and a private key for a set S. Suppose that S satisfies the access structure and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. (Note there could potentially be different ways of choosing the ω_i values to satisfy this.)

The decryption algorithm first computes

$$\begin{aligned} &e(C', K) / \left(\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{\omega_i}\right) \\ &= e(g,g)^{\alpha s} e(g,g)^{ast} / \left(\prod_{i \in I} e(g,g)^{ta\lambda_i \omega_i}\right) \quad = e(g,g)^{\alpha s} \end{aligned}$$

The decryption algorithm can then divide out this value from C and obtain the message $\mathcal{M}$.

3.1 Proof

One important challenge in proving the system secure is that the use of a non-injective ρ function complicates are proof methodology. In particular, in our reduction we want to program our parameters such that for h_x based on the i-th row of M^* if $\rho^*(i) = x$. However, if there exist $i \neq j$ such that $x = \rho(i) = \rho(j)$ then there is an issue since we must program *both* row i and row j in the simulation. Intuitively, there is a potential conflict in how to program the parameters.

To deal with this we apply "parallel" aspect parallel BDHE that gives us "parallel" pieces of the BDHE problem. In our reduction we can use these extra pieces to program all the information in during the simulation.

We prove the following theorem.

Theorem 3.1. *Suppose the decisional q-parallel BDHE assumption holds. Then no polytime adversary can selectively break our system with a challenge matrix of size $\ell^* \times n^*$, where $\ell^*, n^* \leq q$.*

Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\epsilon = \mathsf{Adv}_{\mathcal{A}}$ in the selective security game against our construction. Moreover, suppose it chooses a challenge matrix M^* where both dimensions are at most q. We show how to build a simulator, $\mathcal{B}$, that plays the decisional q-BDHE problem.

Init The simulator takes in a q-parallel BDHE challenge $\vec{y}, T$. The adversary gives the algorithm the challenge access structure (M^*, ρ^*), where M^* has n^* columns.

Setup The simulator chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $e(g,g)^\alpha = e(g^a, g^{a^q}) e(g,g)^{\alpha'}$.

We describe how the simulator "programs" the group elements $h_1, \ldots, h_U$

For each x for $1 \leq x \leq U$ begin by choosing a random value z_x. Let X denote the set of indices i, such that $\rho^*(i) = x$. The simulator programs h_x as:

$$h_x = g^{z_x} \prod_{i \in X} g^{aM_{i,1}/b_i} \cdot g^{a^2 M_{i,2}/b_i} \cdots g^{a^n M_{i,n}/b_i}.$$

Note that if $X = \emptyset$ then we have $h_x = g^{z_x}$. Also note that the parameters are distributed randomly due to the g^{z_x} value.

Phase I In this phase the simulator answers private key queries. Suppose the simulator is given a private key query for a set S where S does not satisfy M^*.

The simulator first chooses a random $r \in \mathbb{Z}_p$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n^*}) \in {\mathbb{Z}_p}^n$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have that $\vec{w} \cdot M_i = 0$. By the definition of a LSSS such a vector must exist.

The simulator begins by implicitly defining t as

$$r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^1$$

. It performs this by setting $L = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{w_i} = g^t$.

We observe that by our definition of t, we have that g^{at} contains a term of $g^{-a^{q+1}}$, which will cancel out with the unknown term in g^α when creating K. The simulator can compute K as:

$$K = g^{\alpha'} g^{ar} \prod_{i=2,\ldots,n^*} (g^{a^{q+2-i}})^{w_i}.$$

Now we must calculate $K_x\ \forall x \in S$. First, we consider $x \in S$ for which there is no i such that $\rho^*(i) = x$. For those we can simply let $K_x = L^{z_x}$.

The more difficult task is to create key components K_x for attributes $x \in S$, where x is used in the access structure. For these keys we must make sure that there are no terms of the form g^{a^{q+1}/b_i} that we can't simulate. However, we have that $M_i \cdot \vec{w} = 0$; therefore, all of these terms cancel.

Again, let X be the set of all i such that $\rho^*(i) = x$. The simulator creates K_x in this case as follows.

$$K_x = L^{z_x} \prod_{i \in X} \prod_{j=1,\ldots,n^*} \left(g^r \prod_{\substack{k=1,\ldots,n^* \\ k \neq j}} (g^{a^{q+1+j-k}/b_i})^{w_k} \right)^{M_{i,j}}$$

Challenge Finally, we build the challenge ciphertext. The adversary gives two messages $\mathcal{M}_0, \mathcal{M}_1$ to the simulator. The simulator flips a coin β. It creates $C = \mathcal{M}_\beta T \cdot e(g^s, g^{\alpha'})$ and $C' = g^s$.

The tricky part is to simulate the C_i values since this contains terms that we must cancel out. However, the simulator can choose the secret splitting, such that these cancel out. Intuitively, the simulator will choose random $y'_2, \ldots, y'_{n^*}$ and the share the secret using the vector

$$\vec{v} = (s, sa + y'_2, sa^2 + y'_3, \ldots, sa^{n-1} + y'_{n^*}) \in {\mathbb{Z}_p}^{n^*}.$$

In addition, it chooses random values $r'_1, \ldots, r'_\ell$.

For $i = 1, \ldots, n^*$, we define R_i as the set of all $k \neq i$ such that $\rho^*(i) = \rho^*(k)$. In other words, the set of all other row indices that have the same attribute as row i. The challenge ciphertext components are then generated as

$$\begin{aligned} D_i &= g^{r'_i} g^{s b_i} \\ C_i &= h_{\rho^*(i)}^{r'_i} \left(\prod_{j=1,\ldots,n^*} (g^a)^{M_{i,j} y_j} \right) (g^{b_i \cdot s})^{-z_{\rho^*(i)}} \left(\prod_{k \in R_i} \prod_{j=1,\ldots,n^*} (g^{a^j \cdot s \cdot (b_i/b_k)})^{M_{k,j}} \right) \end{aligned}$$

Phase II Same as phase I.

Guess The adversary will eventually output a guess β' of β. The simulator then outputs 0 to guesses that $T = e(g,g)^{a^{q+1}s}$ if $\beta = \beta'$; otherwise, it and outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When T is a tuple the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When T is a random group element the message M_β is completely hidden from the adversary and we have $\Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can play the decisional q-parallel BDHE game with non-negligible advantage.

4 Extensions

We discuss extensions and variations of our main construction.

Realizations from Weaker Assumptions One goal is to realize CP-ABE from weaker assumption than the decisional Parallel-BDHE one we introduced. The primary obstacle is to be able to reflect the challenge access structure M^* in the parameters during the reduction. By slightly modifying our system we are able to achieve two other constructions under weaker assumptions.

First, in Appendix A we give a construction provably secure under the existing d-BDHE assumption introduced by Boneh, Boyen and Goh [9]. To accommodate a weaker assumption we introduce a parameter $k_{\max}$ which is the maximum number of times any one attribute can appear in an access formula. Private key in the system will be a factor of $k_{\max}$ larger than our main construction.

In addition, in Appendix B we give a construction provably secure under the much more standard decisional Bilinear Diffie-Hellman assumption. To realize security under this assumption our system must additionally introduce a parameter $n_{\max}$ and encryption and ciphertext sizes will be a factor of n larger than our main construction.

Large Universe of Attributes One aspect of our main construction is that it defines the set of attributes to be used in the parameters. One useful feature is to be able to dynamically use any string as an attribute. In Appendix C we show how in the random oracle we can realize any number of attributes with constant size parameters by simply hashing the attribute string. In Appendix D we provide a large universe construction in the standard model.

Delegation and Chosen-Ciphertext Security We remark that delegation can be realized in essentially the same manner as in the Bethencourt, Sahai, and Waters systems and give a brief sketch. In CP-ABE delegation can be realized by deleting attributes from a key. For instance, if a user has a key for the attribute set {"MANAGER", "ACCOUNTING" } she might like to delegate a key for just the attribute "ACCOUNTING". In our system, to remove an attribute x from a key one needs to transform the key by removing the key component K_x and rerandomize the other key components. To enable this an authority needs to also include g^a in the parameters. Notice that this term was already available in the reduction, thus the security proof is essentially unaffected.

In addition, CCA-security can be realized in the standard model by using the techniques of Canetti, Halevi, and Katz [17] by some simple alternations to the system that apply delegation. In the random oracle model standard methods for realizing CCA-security can be applied.

5 Conclusions

We presented the first ciphertext-policy attribute-based encryption systems that are efficient, expressive, and provably secure under concrete assumptions. All of our constructions fall under a common methodology of embedding an LSSS challenge matrix *directly* into the public parameters. Our constructions provide a tradeoff in terms of efficiency and the complexity of assumptions.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *CRYPTO*, pages 205–222, 2005.

[2] Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In *ICALP (2)*, pages 300–311, 2006.

[3] 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.

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

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

[6] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

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

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

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

[10] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

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

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

[13] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[14] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[15] 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.

[16] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[17] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

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

[19] L. Chen, Keith Harrison, Andrew Moss, David Soldera, and Nigel P. Smart. Certification of public keys within an identity based system. In *ISC*, pages 322–333, 2002.

[20] L. Chen, Keith Harrison, David Soldera, and Nigel P. Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *InfraSec*, pages 260–275, 2002.

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

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

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

[24] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

[25] Vipul Goyal, Abishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute-based encryption. In *ICALP*, 2008.

[26] 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.

[27] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.

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

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

[30] 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.

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

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

[33] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

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

[35] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

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

A Decisional BDHE Construction

We now give our construction that is provably secure under the Bilinear Diffie-Hellman Exponent assumption. We do this in two main steps.

First we give a construction and prove it secure under the restriction that an attribute can only be used in a most one row in the ciphertext access matrix M (the function $\rho(\cdot)$ is injective). This can be thought of as an attribute appearing in at most one node in a formula. Our construction is a small derivative of our main one. Since the ρ function is injective we will not require the extra terms from the Parallel BDHE assumption in order to "program" in the challenge ciphertext.

Second, we address the limitation of letting one element appear at most once with a simple encoding technique. We can simply assign a different string for each time an attribute is associated with a row in an access structure. For example, the attribute "Professor" can be encoded as "Professor:1", "Professor:2", etc. The downside of this transformation is that a users' secret key will grow as $|S| \cdot k_{\max}$, where $k_{\max}$ is the maximum number of times an attribute can be associated with a row in a ciphertext.

For simplicity, we will provide the scheme and proof for when $\rho()$ is an injective function, but note that this simple transformation can allow for attributes to appear at most $k_{\max}$ times with a blowup of a factor of $k_{\max}$ in the private key size. All other performance parameters are identical to our main scheme.

Setup(U) The setup algorithm takes as input the maximum number of system attributes U. Then it chooses a group $\mathbb{G}$ of prime order p and a generator g as well as random group element $h_1, \ldots, h_u \in G$. In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$. The public key is published as

$$\mathrm{PK} = \quad g,\ e(g,g)^{\alpha},\ g^{a},\ h_1, \ldots, h_U.$$

The authority sets $\mathrm{MSK} = g^{\alpha}$ as the master secret key.

Encrypt(PK, $(M,\rho),\mathcal{M}$ **)** The encryption algorithm takes as input the public parameters PK and a message $\mathcal{M}$ to encrypt. In addition, it takes as input an LSSS access structure (M,ρ). The function ρ associates rows of M to attributes. In this construction we limit ρ to be an *injective* function, that is an attribute is associated with at most one row of M.

Let M be an $\ell \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For, $i = 1$ to ℓ it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M.

The ciphertext is published as

$$\mathrm{CT} = \quad \left(C = \mathcal{M}e(g,g)^{\alpha s},\ C' = g^s,\ C_1 = g^{a\lambda_1}h_{\rho(1)}^{-s}, \ldots, C_\ell = g^{a\lambda_\ell}h_{\rho(\ell)}^{-s}\right)$$

along with a description of $M, \rho(\cdot)$.

KeyGen(MSK, S**)** The key generation algorithm takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t \in \mathbb{Z}_p$. It creates the private key as

$$K = g^\alpha g^{at} \quad L = g^t \quad \forall x \in S \quad K_x = h_x^t.$$

Decrypt(CT,SK**)** The decryption algorithm takes as input a ciphertext CT for a linear access structure (M,ρ) and a private key for a set S. Suppose that S satisfies the access structure and let $I \subset \{1,2,\ldots,\ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega \in \mathbb{Z}_p\}_{i\in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i\in I}\omega_i\lambda_i = s$. We note that there could potentially be several different ways of choosing the ω_i values to satisfy this. In addition, we emphasize that the decryption algorithm only needs knowledge of M, I to determine these constants.

The decryption algorithm first computes

$$e(C',K)/\left(\prod_{i\in I}(e(C_i,L)e(C',K_{\rho(i)}))^{\omega_i}\right)$$
$$= e(g,g)^{\alpha s}e(g,g)^{ast}/\left(\prod_{i\in I}e(g,g)^{ta\lambda_i\omega_i}\right) \ = e(g,g)^{\alpha s}.$$

The decryptor can then divide out this value from C and obtain the message $\mathcal{M}$.

A.1 Decisional Bilinear Diffie-Hellman Exponent Assumption

We briefly review the decisional BDHE assumption.

We define the decisional q-Bilinear Diffie-Hellman Exponent problem as follows. Choose a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a, s \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. Let g_i denote g^{a^i}. The adversary when given $\vec{y} = (g, g_1, \ldots, g_q, , g_{q+2}, \ldots, g_{2q}, g^s)$ must distinguish $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving decisional q-BDHE in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}(\vec{y}, T = e(g,g)^{a^{q+1}s}) = 0\right] - \Pr\left[\mathcal{B}(\vec{y}, T = R) = 0\right] \right| \geq \epsilon$$

Definition A.1. *We say that the decisional q-BDHE assumption holds if no polytime algorithm has a non-negligible advantage in solving the (decision) q-BDHE problem.*

A.2 Proof

We prove the following theorem.

Theorem A.2. *Suppose the decisional q-BDHE assumption holds. Then no poly-time adversary can selectively break our system with a challenge matrix of size $\ell^* \times n^*$, where $n^* \leq q$.*

Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\epsilon = \mathsf{Adv}_{\mathcal{A}}$ in the selective security game against our construction. Moreover, suppose it chooses a challenge matrix M^* of dimension at most q columns. We show how to build a simulator, $\mathcal{B}$, that plays the decisional q-BDHE problem.

Init The simulator takes in the BDHE challenge $\vec{y} = (g, g^s, g^a, \ldots, g^{a^q}, , g^{a^{q+2}} \ldots, g^{a^{2q}}), T$. The adversary gives the algorithm the challenge access structure (M^*, ρ^*), where M^* has $n^* \leq q$ columns.

Setup The simulator chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $e(g,g)^\alpha = e(g^a, g^{a^q})e(g,g)^{\alpha'}$.

We describe how the simulator program the parameters $h_1, \ldots, h_U$. For each x from 1 to U choose a random value $z_x \in \mathbb{Z}_p$. If there exists an i such that $\rho^*(i) = x$, then let

$$h_x = g^{z_x} g^{aM^*_{i,1}} \cdot g^{a^2 M^*_{i,2}} \cdots g^{a^{n^*} M^*_{i,n^*}}.$$

Otherwise, let $h_x = g^{z_x}$.

We point out a couple of facts. First, the parameters are distributed randomly due to the g^{z_x} factor. Second, by our restriction that ρ^* to be an injective function for any x there is at most one i such that $\rho^*(i) = x$; therefore, our assignment is unambiguous.

Intuitively, for each attribute x that is represented in the challenge ciphertext we are able to program the parameter h_x to reflect the corresponding row M^* from the simulator's point of view. We see that the $q \geq n^*$ terms from the assumption enable us to represent this in just one group element h_x without ambiguity.

Phase I In this phase the simulator answers private key queries. Suppose the simulator is given a private key query for a set S where S does not satisfy M^*.

The simulator first chooses a random $r \in \mathbb{Z}_p$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n^*}) \in \mathbb{Z}_p{}^{n^*}$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have that $\vec{w} \cdot M^*_i = 0$. By our the definition of a LSSS such a vector must exist, since S does not satisfy M^*.

The simulator begins by implicitly defining t as:

$$r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^{q-n^*+1}.$$

It performs this by setting $L = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{w_i} = g^t$.

We now observe that by our definition of t, we have that g^{at} contains a term of $g^{-a^{q+1}}$, which will cancel out with the unknown term in g^α. The simulator can compute K as:

$$K = g^{\alpha'} g^{ar} \prod_{i=2,\ldots,n^*} (g^{a^{q+2-i}})^{w_i}.$$

Now we must calculate $K_x \; \forall x \in S$. First, we consider $x \in S$ for which there is no i such that $\rho^*(i) = x$. For those we can simply let $K_x = L^{z_x}$.

The more difficult task is to create keys for attributes x, where x is used in the access structure. For these keys we must make sure that there are no terms of the form $g^{a^{q+1}}$ that we cannot simulate. Notice, that in calculating hx^t all terms of this form come (in the exponent) from $M_{i,j}a^j \cdot w_j a^{q+1-j}$ for some j, where $\rho^*(i) = x$. However, we have that $M_i \cdot \vec{w} = 0$; therefore, everything with an exponent of a^{q+1} cancels when combined.

The simulator creates K_x in this case as follows. Suppose $\rho^*(i) = x$. Then

$$K_x = L^{z_x} \prod_{j=1,\ldots,n^*} \left(g^r \prod_{\substack{k=1,\ldots,n^* \\ k \neq j}} (g^{a^{q+1+j-k}})^{w_k} \right)^{M_{i,j}} .$$

Challenge Finally, we build the challenge ciphertext. The adversary gives two messages $\mathcal{M}_0, \mathcal{M}_1$ to the simulator. The simulator flips a coin β. It creates $C = \mathcal{M}_\beta T \cdot e(g^s, g^{\alpha'})$ and $C' = g^s$.

The tricky part is to simulate the C_i values since the term $h^s_{\rho^*(i)}$ will contain terms of the form $g^{a^j s}$, that we do not know how to simulate. However, the simulator can choose the secret splitting, such that these cancel out. Intuitively, the simulator will choose random $y'_2, \ldots, y'_{n^*} \in \mathbb{Z}_p$ and the share the secret using the vector

$$\vec{v} = (s, sa + y'_2, sa^2 + y'_3, \ldots, sa^{n^*-1} + y'_{n^*}) \in \mathbb{Z}_p{}^{n^*}.$$

This allows the important terms from $h^{-s}_{\rho(i)}$ to cancel out with the important terms of $g^{a\lambda_i}$. For $i = 1, \ldots, n^*$ the challenge ciphertext components are then generated as

$$C_i = \left(\prod_{j=1,\ldots,n^*} (g^a)^{M_{i,j} y'_j} \right) (g^s)^{-z_{\rho^*(i)}}.$$

Phase II Same as phase I.

Guess The adversary will eventually output a guess β' of β. The simulator then outputs 0 to guesses that $T = e(g,g)^{a^{q+1}s}$ if $\beta = \beta'$; otherwise, it and outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When T is a tuple the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When T is a random group element the message M_β is completely hidden from the adversary and we have $\Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can play the decisional q-BDHE game with non-negligible advantage.

A.3 Necessity of injectiveness in $\rho()$ function

The current construction states that the $\rho()$ function that maps rows of a ciphertext access matrix M to attributes must be injective. This implies that an attribute cannot appear twice in an access structure. One natural question is whether this is an inherent requirement of the construction or whether it is an artifact of the proof provided.

We show an example of an attack when the $\rho()$ function is not injective. Suppose the a ciphertext matrix M has a function ρ that maps the first four row as

$$\rho(1) = \rho(2) = \text{“Attribute X”} \quad \rho(3) = \rho(4) = \text{“Attribute Y”}.$$

Suppose that after randomly choosing $\vec{v} = (s, y_2, \ldots, y_n)$ the first four shares of the ciphertext were

$$\lambda_1 = 2s + y_2 \ , \lambda_2 = s + y_3 \ , \lambda_3 = -y_2 \ , \lambda_4 = -y_3.$$

Then an attacker can compute $C_1 \cdot C_2^{-1} \cdot C_3 \cdot C_4^{-1} = g^{as}$. Then he can use a key that does not satisfy the access structure to decrypt by computing $e(K, C')/e(L, g^{as}) = e(g, g)^{\alpha \cdot s}$.

Due to this attack, the limitation of ρ to be injective is inherent in this construction. As we stated earlier we can apply a simply naming transformation that works around around this issue. With increasing the private key size by a factor of $k_{\max}$ we can allow for an attribute to be used $k_{\max}$ times. However, in the next section we provide a construction that gets around this issue altogether with no performance penalty.

B Bilinear Diffie-Hellman Construction

While our unrestricted construction realizes a potentially ideal type of efficiency, we would like to also show that secure CP-ABE systems can be realized from static assumptions. Here we show how to realize our framework under the decisional Bilinear Diffie Hellman d-(BDH) assumption.

The primary challenge with realizing a construction provably secure under BDH is we need a way for a reduction to embed the challenge matrix M^* in the parameters. Since the BDH assumption gives the reduction less components to embed this, there is no obvious path for reducing the previous constructions to d-BDH. We surmount this obstacle by expanding our ciphertexts and public parameter space. By doing this we enable our reduction to embed the challenge matrix.

Our construction is parametrized by a integer $n_{\max}$ that specifies the maximum number of columns in a ciphertext. Like our first construction we restrict $\rho()$ to be an injective functions, but can alleviate this restriction by applying a similar transformation to allow an attribute to appear $k_{\max}$ times for some specified $k_{\max}$. Our construction follows.

Setup($U, n_{\max}$) The setup algorithm takes as input, U, the number of attributes in the system U and $n_{\max}$ the maximum number of columns in an LSSS matrix (or number of nodes in an access formula). It then creates a group $\mathbb{G}$ of prime order p and a generator g and chooses random element $(h_{1,1}, \ldots, h_{1,U}), \ldots, (h_{n_{\max},1}, \ldots, h_{n_{\max},U})$ In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$.

The public key is published as

$$\text{PK} = \quad g, \ e(g,g)^{\alpha}, \ g^{a} \ (h_{1,1}, \ldots, h_{1,U}), \ldots, (h_{n_{\max},1}, \ldots, h_{n_{\max},U})$$

The authority sets MSK $= g^{\alpha}$ as the master secret key.

Encrypt(PK, $(M,\rho), \mathcal{M}$) The encryption algorithm takes as input the public parameters PK and a message $\mathcal{M}$ to encrypt. In addition, it takes as input an LSSS access structure (M,ρ). The function ρ associates rows of M to attributes. In this construction we limit ρ to be an injective function, that is an attribute is associated with at most one row of M.

Let M be an $\ell \times n$ matrix. Note n may be less than or equal to $n_{\max}$. The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s.

The ciphertext is published as

$$\mathrm{CT} = C = \mathcal{M} e(g,g)^{\alpha s},\ C' = g^s,\ \forall_{\substack{i=1,\ldots,\ell \\ j=1,\ldots,n}} C_{i,j} = g^{aM_{i,j}v_j} h_{j,\rho(i)}^{-s}$$

along with a description of M, ρ.

KeyGen(MSK, S) The key generation algorithm takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t_1, \ldots, t_{n_{\max}} \in \mathbb{Z}_p$. It creates the private key as

$$K = g^{\alpha} g^{a t_1} \quad L_1 = g^{t_1}, \ldots, L_n = g^{t_{n_{\max}}} \quad \forall x \in S \quad K_x = \prod_{j=1,\ldots,n_{\max}} h_{j,x}^{t_j}.$$

Decrypt(CT,SK) The decryption algorithm takes as input a ciphertext CT for access structure (M,ρ) and a private key for a set S. Suppose that S satisfies the access structure and let $I \subset \{1,2,\ldots,\ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that, if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. (Note there could potentially be different ways of choosing the ω_i values to satisfy this.)

The decryption algorithm first computes

$$\begin{aligned}
& e(C',K) / \left(\prod_{j=1,\ldots,n} e(L_j, \prod_{i \in I} C_{i,j}^{\omega_i}) \right) \prod_{i \in I} e(K_{\rho(i)}^{\omega_i}, C') \\
= \quad & e(C',K) / \left(\prod_{j=1,\ldots,n} e(g^{t_j}, g^{\sum_{i \in I} a M_{i,j} v_j \omega_i}) \cdot e(g^{t_j}, \prod_{i \in I} h_{j,\rho(i)}^{-s\omega_i}) \right) \prod_{i \in I} e(K_{\rho(i)}^{\omega_i}, g^s) \\
= \quad & e(C',K) / \prod_{j=1,\ldots,n} e(g^{t_j}, g^{\sum_{i \in I} a M_{i,j} v_j \omega_i}) \\
= \quad & e(C',K) / e(g^{t_1}, g^{\sum_{i \in I} a M_{i,1} v_1 \omega_i}) \\
= \quad & e(g^s, g^{\alpha} g^{a t_1}) / e(g,g)^{a t_1 s} \\
= \quad & e(g,g)^{\alpha s}
\end{aligned}$$

The decryptor can then divide out this value from C and obtain the message $\mathcal{M}$.

B.1 Decisional Bilinear Diffie-Hellman Assumption

We briefly review the Decisional Bilinear Diffie-Hellman Assumption.

We define the decisional Bilinear Diffie-Hellman problem as follows. A challenger chooses a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a, b, s \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. The adversary when given (g, g^a, g^b, g^s) must distinguish a valid tuple $e(g,g)^{abs} \in \mathbb{G}_T$ from a random element R in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving decisional BDH in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\big(g, g^a, g^b, g^s, T = e(g,g)^{abs}\big) = 0\right] - \Pr\left[\mathcal{B}\big(g, g^a, g^b, g^s, T = R\big) = 0\right] \right| \geq \epsilon$$

Definition B.1. *We say that the decisional BDH assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional BDH problem.*

B.2 Proof

We prove the following theorem.

Theorem B.2. *Suppose the decisional BDH assumption holds. Then no polytime adversary can selectively break our system.*

Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\epsilon = \mathsf{Adv}_{\mathcal{A}}$ in the selective security game against our construction. We show how to build a simulator, $\mathcal{B}$, that plays the decisional BDH problem.

Init The simulator takes in the BDH challenge g, g^a, g^b, g^s, g^a, T. The adversary gives the algorithm the challenge access structure (M^*, ρ^*), where M^* has n^* columns. The adversary also specifies an $n_{\max} \geq n^*$.

Setup The simulator chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = ab + \alpha'$ by letting $e(g,g)^\alpha = e(g^a, g^b)e(g,g)^{\alpha'}$.

Next, we describe how the simulator programs the parameters $(h_{1,1}, \ldots, h_{1,U}), \ldots, (h_{n_{\max},1}, \ldots, h_{n_{\max},U})$.

For each j, x pair where $1 \leq x \leq U$ and $1 \leq j \leq n_{\max}$ it choose a random value $z_{x,j} \in \mathbb{Z}_p$. If there exists an i such that $\rho^*(i) = x$ and $i \leq n^*$, then let

$$h_{j,x} = g^{z_{x,j}} g^{aM_{i,j}}$$

Otherwise, let $h_x = g^{z_{x,j}}$.

We point out a couple of facts. First, the public parameters distributed randomly due to the g^{z_x} value. Second, by our restriction for any x there is at most one i such that $\rho^*(i) = x$, therefore our assignment is unambiguous.

Phase I In this phase the simulator answers private key queries. Suppose the simulator is given a private key query for a set S where S does not satisfy M^*.

The simulator first chooses a random $r_1, \ldots, r_{n_{\max}} \in \mathbb{Z}_p$. For this discussion we define $M_{i,j} = 0$ for $n^* < j \leq n_{\max}$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n_{\max}}) \in {\mathbb{Z}_p}^{n_{\max}}$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have that $\vec{W} \cdot M_i = 0$. By the definition of a LSSS such a vector must exist. Note we can simply let $w_j = 0$ and consider $M_{i,j} = 0$ for $n^* < j \leq n_{\max}$.

The simulator begins by implicitly defining t_j as

$$r_j + w_j \cdot b$$

.

It can then calculate all L_j as $L_j = g^{r_j}(g^b)^{w_j}$

We now observe that by our definition of t_1, we have that g^{at_1} contains a term of g^{ab}, which will cancel out with the unknown term in g^{α}. The simulator can compute K as:

$$K = g^{\alpha'} g^{ar_1}.$$

Now we must calculate K_x $\forall x \in S$. First, we consider $x \in S$ for which there is no i such that $\rho^*(i) = x$. For those we can simply let $K_x = \prod_{j=1,\ldots,n} L_j^{z_{x,j}}$.

The more difficult task is to create keys for attributes x, where x is used in the access structure. For these keys we must make sure that there are no terms of the form g^{ab} that we can't simulate. Notice, that in calculating $\prod_j h_{j,x}$ all terms of this form come from $M_{i,j}a \cdot w_j b$ for some j, where $\rho^*(i) = x$. However, we have that $M_i \cdot \vec{w} = 0$; therefore, all of these terms cancel.

The simulator creates K_x in this case as

$$K_x = \prod_{j=1,\ldots,n_{\max}} g^{z_{x,j}r_j} \cdot g^{bz_{x,j}} \cdot g^{aM_{i,j}r_j}.$$

Challenge Finally, we build the challenge ciphertext. The adversary gives two messages $\mathcal{M}_0, \mathcal{M}_1$ to the simulator. The simulator flips a coin β. It creates $C = \mathcal{M}_\beta T e(g^s, g^{\alpha'})$ and $C' = g^s$.

The tricky part is to simulate the $C_{i,j}$ values since the term $h^s_{j,\rho^*(i)}$ will contain terms of the form g^{as}, that we do not know how to simulate. However, the simulator can choose the secret splitting, such that these cancel out. Intuitively, the simulator will choose random $y_2; , \ldots, y'_{n^*}$ and the share the secret using the vector

$$\vec{v} = (s, s + y'_2, s + y'_3, \ldots, s + y'_{n^*}) \in \mathbb{Z}_p{}^{n^*}.$$

For $i = 1, \ldots, n$ the challenge ciphertext components are then generated as

$$C_{i,j} = (g^a)^{M_{i,j}y_j}(g^s)^{-z_{\rho^*(i),j}}.$$

Phase II Same as Phase I.

Guess The adversary will eventually output a guess β' of β. The simulator then outputs 0 to guesses that $T = e(g,g)^{abs}$ if $\beta = \beta'$; otherwise, it and outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When T is a tuple the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{abs}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When T is a random group element the message M_β is completely hidden from the adversary and we have $\Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can play the decisional BDH game with non-negligible advantage.

C Large Universe Construction

Our main construction was limited to the "small-universe" case where the set of attributes $\mathcal{U}$ is defined at system setup and the size of the public parameters grows with $|\mathcal{U}|$. In this section, we show a variation of the scheme where the number of attributes is unlimited and the public parameter size is constant. The construction is a simple adaptation of the main where an attribute parameter can be dynamically derived by hashing the attribute with a function $H : \{0,1\}^* \rightarrow \mathbb{G}$ that we model as a random oracle.

Setup() The setup algorithm chooses a group $\mathbb{G}$ of prime order p and a generator g. In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$. In addition, we will use a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ that we will model as a random oracle. The public key is published as

$$\text{PK} = \quad g,\ e(g,g)^{\alpha},\ g^{a}.$$

The authority sets $\text{MSK} = g^{\alpha}$ as the master secret key.

Encrypt($\text{PK}, (M,\rho), \mathcal{M}$ **)** The encryption algorithm takes as input the public parameters PK and a message $\mathcal{M}$ to encrypt. In addition, it takes as input an LSSS access structure (M,ρ). The function ρ associates rows of M to attributes. In this construction we limit ρ to be an injective function, that is an attribute is associated with at most one row of M.

Let M be an $\ell \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For $i = 1$ to ℓ, it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. In addition, the algorithm chooses random $r_1, \ldots, r_\ell \in \mathbb{Z}_p$.

The ciphertext is published as CT =

$$C = \mathcal{M}e(g,g)^{\alpha s},\ C' = g^s, (C_1 = g^{a\lambda_1}H(\rho(1))^{-r_1},\ D_1 = g^{r_1}), \ldots, (C_n = g^{a\lambda_n}H(\rho(n))^{-r_n},\ D_n = g^{r_n})$$

along with a description of (M,ρ).

KeyGen(MSK, S**)** The key generation algorithm takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t \in \mathbb{Z}_p$. It creates the private key as

$$K = g^{\alpha}g^{at} \quad L = g^t \quad \forall x \in S\ K_x = H(x)^t.$$

We remark that key generation is actually the same as in our previous construction.

Decrypt(CT,SK) The decryption algorithm takes as input a ciphertext CT for access structure (M,ρ) and a private key for a set S. Suppose that S satisfies the access structure and let $I \subset \{1,2,\ldots,\ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega \in \mathbb{Z}_p\}_{i\in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i\in I}\omega_i\lambda_i = s$. (Note there could potentially be different ways of choosing the ω_i values to satisfy this.)

The decryption algorithm first computes

$$\begin{aligned} &e(C',K)/\left(\prod_{i\in I}(e(C_i,L)e(D_i,K_{\rho(i)}))^{\omega_i}\right) \\ &= e(g,g)^{\alpha s}e(g,g)^{ast}/\left(\prod_{i\in I}e(g,g)^{ta\lambda_i\omega_i}\right) \quad = e(g,g)^{\alpha s} \end{aligned}$$

The decryption algorithm can then divide out this value from C and obtain the message $\mathcal{M}$.

C.1 Proof

We prove the following theorem.

Theorem C.1. *Suppose the decisional q-parallel BDHE assumption holds. Then no polytime adversary can selectively break our system with a challenge matrix of size $\ell^* \times n^*$, where $\ell^*, n^* \leq q$.*

Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\epsilon = \mathsf{Adv}_{\mathcal{A}}$ in the selective security game against our construction. Moreover, suppose it chooses a challenge matrix M^* where both dimensions are at most q. We show how to build a simulator, $\mathcal{B}$, that plays the decisional q-BDHE problem.

Init The simulator takes in a q-parallel BDHE challenge $\vec{y}, T$. The adversary gives the algorithm the challenge access structure (M^*, ρ^*), where M^* has n^* columns.

Setup The simulator chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $e(g,g)^\alpha = e(g^a, g^{a^q})e(g,g)^{\alpha'}$.

We describe how the simulator programs the random oracle H by building a table. Consider a call to $H(x)$. If $H(x)$ was already defined in the table, then simply return the same answer as before.

Otherwise, begin by choosing a random value z_x. Let X denote the set of indices i, such that $\rho^*(i) = x$. The simulator programs the oracle as

$$H(x) = g^{z_x} \prod_{i \in X} g^{aM_{i,1}/b_i} \cdot g^{a^2 M_{i,2}/b_i} \cdots g^{a^n M_{i,n}/b_i}.$$

Note that if $X = \emptyset$ then we have $H(x) = g^{z_x}$. Also note that the responses from the oracle are distributed randomly due to the g^{z_x} value.

Phase I In this phase the simulator answers private key queries. Suppose the simulator is given a private key query for a set S where S does not satisfy M^*.

The simulator first chooses a random $r \in \mathbb{Z}_p$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n^*}) \in {\mathbb{Z}_p}^n$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have that $\vec{W} \cdot M_i = 0$. By the definition of a LSSS such a vector must exist.

The simulator begins by implicitly defining t as

$$r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^1$$

. It performs this by setting $L = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{w_i} = g^t$.

We observe that by our definition of t, we have that g^{at} contains a term of $g^{-a^{q+1}}$, which will cancel out with the unknown term in g^α when creating K. The simulator can compute K as:

$$K = g^{\alpha'} g^{ar} \prod_{i=2,\ldots,n^*} (g^{a^{q+2-i}})^{w_i}.$$

Now we must calculate K_x $\forall x \in S$. First, we consider $x \in S$ for which there is no i such that $\rho^*(i) = x$. For those we can simply let $K_x = L^{z_x}$.

The more difficult task is to create key components K_x for attributes $x \in S$, where x is used in the access structure. For these keys we must make sure that there are no terms of the form g^{a^{q+1}/b_i} that we can't simulate. However, we have that $M_i \cdot \vec{w} = 0$; therefore, all of these terms cancel.

Again, let X be the set of all i such that $\rho^*(i) = x$. The simulator creates K_x in this case as follows.

$$K_x = L^{z_x} \prod_{i \in X} \prod_{j=1,\ldots,n^*} \left(g^r \prod_{\substack{k=1,\ldots,n^* \\ k \neq j}} (g^{a^{q+1+j-k}/b_i})^{w_k} \right)^{M_{i,j}}$$

Challenge Finally, we build the challenge ciphertext. The adversary gives two messages $\mathcal{M}_0, \mathcal{M}_1$ to the simulator. The simulator flips a coin β. It creates $C = \mathcal{M}_\beta T \cdot e(g^s, g^{\alpha'})$ and $C' = g^s$.

The tricky part is to simulate the C_i values since this contains terms that we must cancel out. However, the simulator can choose the secret splitting, such that these cancel out. Intuitively, the simulator will choose random $y'_2, \ldots, y'_{n^*}$ and the share the secret using the vector

$$\vec{v} = (s, sa + y'_2, sa^2 + y'_3, \ldots, sa^{n-1} + y'_{n^*}) \in \mathbb{Z}_p{}^{n^*}.$$

In addition, it chooses random values $r'_1, \ldots, r'_\ell$.

For $i = 1, \ldots, n^*$, we define R_i as the set of all $k \neq i$ such that $\rho^*(i) = \rho^*(k)$. In other words, the set of all other row indices that have the same attribute as row i. The challenge ciphertext components are then generated as

$$\begin{aligned} D_i &= g^{r'_i} g^{sb_i} \\ C_i &= H(\rho^*(i))^{r'_i} \left(\prod_{j=1,\ldots,n^*} (g^a)^{M_{i,j} y_j} \right) (g^{b_i \cdot s})^{-z_{\rho^*(i)}} \left(\prod_{k \in R_i} \prod_{j=1,\ldots,n^*} (g^{a^j \cdot s \cdot (b_i/b_k)})^{M_{k,j}} \right) \end{aligned}$$

Phase II Same as phase I.

Guess The adversary will eventually output a guess β' of β. The simulator then outputs 0 to guesses that $T = e(g,g)^{a^{q+1}s}$ if $\beta = \beta'$; otherwise, it and outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When T is a tuple the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When T is a random group element the message M_β is completely hidden from the adversary and we have $\Pr[\mathcal{B}(\vec{y}, T = R) = 0] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can play the decisional q-parallel BDHE game with non-negligible advantage.

D Removing the Random Oracle for Large Universes of Attributes

In this section we describe how to realize a large universe construction in the standard model. In Appendix C the reduction applied the random oracle to "program" in the challenge ciphertext. the hash function. In the standard model we can achieve a similar type of programming by simply using a hash function that has enough degrees of randomness to plug in the same information. The tradeoff is that the system must define at setup, $\mathit{Attr}_{\max}$, the maximum number of attributes any one key may have and the public paramters grow linearly with $\mathit{Attr}_{\max}$. We emphasize that $\mathit{Attr}_{\max}$ does not limit the number of attributes that may be used in the system.

We realize our standard model construction actually by adapting the construction from Appendix A. We then prove it secure under the decisional BDHE assumption in the standard model. We remark that similar techniques can be used to a realize large universe variant of our main constructions in the standard model, although we do not provide the details here.

Setup($\mathit{Attr}_{\max}$**,**$\ell_{\max}$**)** The setup algorithm takes as inputs the maximum number of attributes a user's key may have and the maximum number of columns in a ciphertext access matrix. The setup algorithm chooses a group $\mathbb{G}$ of prime order p and a generator g. For ease of exposition, we will assume attributes can be represented in $\mathbb{Z}_p$. In practice, one would simply apply a collision resistant hash function from

The algorithm it chooses random exponents $\alpha, a \in \mathbb{Z}_p$. In addition, we will define a hash function $U : \mathbb{Z}_p \to \mathbb{G}$. It does this by implicitly choosing a polynomial $p(x) \in \mathbb{Z}_p$ of degree $m = \mathit{Attr}_{\max} + \ell_{\max} - 1$. Next it computes $u_0 = g^{p(0)}, \ldots, u_m = g^{p(m)}$. These $m + 1$ values when published will allow anyone (by interpolation) to compute $g^{p(x)}$ for any $x \in \mathbb{Z}_p$. The public key is published as

$$\mathrm{PK} = \quad g,\ e(g,g)^{\alpha},\ g^{a},\ u_0, \ldots, u_m.$$

The authority sets $\mathrm{MSK} = g^{\alpha}$ as the master secret key.

Encrypt($\mathrm{PK}, (M,\rho), \mathcal{M}$ **)** The encryption algorithm takes as input the public parameters PK and a message $\mathcal{M}$ to encrypt. In addition, it takes as input an LSSS access structure (M, ρ). The function ρ associates rows of M to attributes. In this construction we limit ρ to be an *injective* function, that is an attribute is associated with at most one row of M. That any attribute is associated with at most one row.

Let M be an $\ell \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For, $i = 1$ to ℓ it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M.

The ciphertext is published as

$$\mathrm{CT} = \quad \left(C = \mathcal{M}e(g,g)^{\alpha s},\ C' = g^{s},\ C_1 = g^{a\lambda_1}U(\rho(1))^{-s}, \ldots, C_\ell = g^{a\lambda_\ell}U(\rho(\ell))^{-s}\right)$$

along with a description of $M, \rho(\cdot)$.

KeyGen(MSK, S**)** The key generation algorithm takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t \in \mathbb{Z}_p$. It creates the private key as

$$K = g^{\alpha} g^{at} \quad L = g^{t} \quad \forall x \in S \quad K_x = U(x)^{t}.$$

Decrypt(CT,SK**)** The decryption algorithm takes as input a ciphertext CT for a linear access structure (M, ρ) and a private key for a set S. Suppose that S satisfies the access structure and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. (Note that there could potentially be several different ways of choosing the ω_i values to satisfy this.)

The decryption algorithm first computes

$$e(C', K) / \left(\prod_{i \in I} (e(C_i, L) e(C', K_{\rho(i)}))^{\omega_i} \right)$$
$$= e(g,g)^{\alpha s} e(g,g)^{ast} / \left(\prod_{i \in I} e(g,g)^{ta\lambda_i \omega_i} \right) = e(g,g)^{\alpha s}.$$

The decryptor can then divide out this value from C and obtain the message $\mathcal{M}$.

D.1 Proof

We prove the following theorem.

Theorem D.1. *Suppose the decisional q-BDHE assumption holds. Then no poly-time adversary can selectively break our system with a challenge matrix of size $\ell^* \times n^*$ and maximum number of attributes per key of $Attr_{\max}$ where $n^* + Attr_{\max} \leq q$.*

Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\epsilon = \mathsf{Adv}_{\mathcal{A}}$ in the selective security game against our construction. Moreover, suppose it chooses a challenge matrix M^* of dimension at most q columns. We show how to build a simulator, $\mathcal{B}$, that plays the decisional q-BDHE problem.

Init The simulator takes in the BDHE challenge $\vec{y} = (g, g^s, g^a, \ldots, g^{a^q}, , g^{a^{q+2}} \ldots, g^{a^{2q}}), T$. The adversary gives the algorithm the challenge access structure (M^*, ρ^*), where M^* has n^* columns and $n^* + Attr_{\max} \leq q$.

Setup The simulator chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $e(g,g)^\alpha = e(g^a, g^{a^q}) e(g,g)^{\alpha'}$.

We describe how the simulator programs the function $U = p(x)$. It chooses $n^* + Attr_{\max} + 1$ polynomials $p_0, p_1, \ldots, p_{n^* + Attr_{\max}}$ each of degree $Attr_{\max} + \ell^*$. Polynomial p_0 is chosen randomly. It lets polynomials $p_{n^*+1}, \ldots, p_{n^* + Attr_{\max}}$ be set to 0 for the ℓ^* values of x where we there exists an i such that $\rho^*(i) = x$ and random elsewhere. The polynomials $p_1, \ldots, p_{n^*}$ are set such that for each x such that there exists an i where $x = \rho^*(i)$ we set $p_j(x) = M^*_{i,j}$ for $j \in [1, n^*]$.

We then conceptually will set

$$p(x) = \sum_{j \in [0, n^* + Attr_{\max}]} p_j(x) \cdot \alpha^j.$$

Using interpolation and the powers g^{a^j} from the assumption we can compute $u_0, \ldots, u_{Attr_{\max} + n_{\max}}$.

We point out a couple of facts. First, the parameters are distributed randomly due to the p_0 component of the polynomial. Second, by our restriction that ρ^* to be an injective function for any x there is at most one i such that $\rho^*(i) = x$; therefore, our assignment is well defined.

Intuitively, for each attribute x that is represented in the challenge ciphertext we are able to program the function $U(x)$ to reflect the corresponding row M^* from the simulator's point of view.

Phase I In this phase the simulator answers private key queries. Suppose the simulator is given a private key query for a set S where S does not satisfy M^*.

The simulator first chooses a random $r \in \mathbb{Z}_p$. Next, define the vector $\vec{b_x}$ such that $b_{x,j} = p_j(x)$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n^*+Attr_{\max}}) \in {\mathbb{Z}_p}^{n^*}$ such that $w_1 = -1$ and for all $x \in S$ we have $\vec{b_x} \cdot \vec{w} = 0$. By our the definition of a LSSS such a vector must exist, since S does not satisfy M^*. The first components $w_1, \ldots, w_{n^*}$ are chosen as before to satisfy all $x \in S$ where x is an attribute in the challenge set. The other vector components $w_{n^*+1} + w_{n^*+Attr_{\max}}$ can be chosen to satisfy the orthogonality condition for all other x with high probability. Note this is possible since the polynomials are of degree $Attr_{\max} + n^*$.

The simulator begins by implicitly defining t as:

$$r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{(n^*+Attr_{\max})} \cdot a^{q-n^*+Attr_{\max}+1}.$$

It performs this by setting $L = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{w_i} = g^t$.

We now observe that by our definition of t, we have that g^{at} contains a term of $g^{-a^{q+1}}$, which will cancel out with the unknown term in g^{α}. The simulator can compute K as:

$$K = g^{\alpha'} g^{ar} \prod_{i=2,\ldots,(n^*+Attr_{\max})} (g^{a^{q+2-i}})^{w_i}.$$

Now we must calculate K_x $\forall x \in S$. For these key components we must make sure that there are no terms of the form $g^{a^{q+1}}$ that we cannot simulate. Notice, that in calculating $H(x)^t$ all terms of this form come (in the exponent) from $M_{i,j} a^j \cdot w_j a^{q+1-j}$ for some j, where $\rho^*(i) = x$. However, we have that $M_i \cdot \vec{w} = 0$; therefore, everything with an exponent of a^{q+1} cancels when combined.

The simulator creates K_x in this case as follows. Suppose $\rho^*(i) = x$. Then

$$K_x = L^{p_0(x)} \prod_{j=1,\ldots,n^*+Attr_{\max}} \left(g^r \prod_{\substack{k=1,\ldots,n^*+Attr_{\max} \\ k \neq j}} (g^{a^{q+1+j-k}})^{w_k} \right)^{p_j(x)}.$$

Challenge Finally, we build the challenge ciphertext. The adversary gives two messages $\mathcal{M}_0, \mathcal{M}_1$ to the simulator. The simulator flips a coin β. It creates $C = \mathcal{M}_\beta \cdot T \cdot e(g^s, g^{\alpha'})$ and $C' = g^s$.

The tricky part is to simulate the C_i values since the term $H(\rho^*(i))^s$ will contain terms of the form $g^{a^j s}$, that we do not know how to simulate. However, the simulator can choose the secret splitting, such that these cancel out. Intuitively, the simulator will choose random $y'_2, \ldots, y'_{n^*} \in \mathbb{Z}_p$ and the share the secret using the vector

$$\vec{v} = (s, sa + y'_2, sa^2 + y'_3, \ldots, sa^{n^*-1} + y'_{n^*}) \in {\mathbb{Z}_p}^{n^*}.$$

This allows the important terms from $H(\rho(i))^{-s}$ to cancel out with the important terms of $g^{a\lambda_i}$. For $i = 1, \ldots, n^*$ the challenge ciphertext components are then generated as

$$C_i = \left(\prod_{j=1,\ldots,n^*} (g^a)^{M_{i,j} y'_j} \right) (g^s)^{p_0(\rho^*(i))}.$$

Phase II Same as phase I.

Guess The adversary will eventually output a guess β' of β. The simulator then outputs 0 to guesses that $T = e(g,g)^{a^{q+1}s}$ if $\beta = \beta'$; otherwise, it and outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When T is a tuple the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When T is a random group element the message M_β is completely hidden from the adversary and we have $\Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can play the decisional n^*-BDHE game with non-negligible advantage.

E Generic Security of Parallel BDHE

We briefly show that the decisional parallel-BDHE assumption is generically secure. We use the generic proof template of Boneh, Boyen, and Goh [9].

We first recall the decisional q-parallel Bilinear Diffie-Hellman Exponent problem from Section 2. Choose a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a, s \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. If an adversary is given $\vec{y}$=

$$\begin{array}{rc}
 & g, g^s, g^a, \ldots, g^{(a^q)},\ , g^{(a^{q+2})}, \ldots, g^{(a^{2q})} \\
\forall_{1\le j\le q} & g^{s\cdot b_j},\quad g^{a/b_j}, \ldots, g^{(a^q/b_j)},\ , g^{(a^{q+2}/b_j)}, \ldots, g^{(a^{2q}/b_j)} \\
\forall_{1\le j,k\le q, k\ne j} & g^{a\cdot s\cdot b_k/b_j}, \ldots, g^{(a^q\cdot s\cdot b_k/b_j)}
\end{array}$$

it must remain hard to distinguish $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

Using the terminology from BBG we need to show that $f = a^{q+1}s$ in independent of the polynomials P and Q. Since all given terms are in the bilinear group we have that $Q = \{1\}$ and we have that

$$P = \{1, s,\ \forall_{i\in[1,2q], j,k\in[1,q], i\ne q+1, j\ne k}\quad a^i,\ a^i/b_j,\ s\cdot b_j,\ a^i\cdot s\cdot b_k/b_j\}$$

We first note that this case at first might appear to be outside the BBG framework, since the polynomials are rational function (due to the b_j^{-1} terms. However, by a simple renaming of terms we can see this is equivalent to an assumption where we use a generator u and let $g = g^{\prod_{j\in[1,q]} b_j}$. Applying this substitution we get a a set of polynomials where maximum degree of any polynomial in the set P is $3q+1$.

We need to also check that f is symbolically independent of the of any two polynomials in P, Q. To realize f from P, Q we would need to have a term of the form $a^{m+1}s$. We note that no such terms can be realized from the product of two polynomials $p, p' \in P$. To form such a term it would need to have a single factor of s. However, if we use the polynomial s as p then no other potential p' has $a^m + 1$. If we use $a^i s \cdot b_k/b_j$ as p, then no matter what p' is used there will be a factor of b_j left. It follows from the BBG framework that this is generically secure.