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

Brent Waters*
SRI International

Abstract

We present new techniques for realizing Ciphertext-Policy Attribute Encryption (CP-ABE) under concrete and noninteractive cryptographic assumptions. Our solutions allow any encryptor to specify access control in terms of an LSSS matrix, M, over the attributes in the system. We present three different constructions that allow different tradeoffs between the systems efficiency and the complexity of the assumptions used. All three constructions use a common methodology of "directly" solving the CP-ABE problem that enable us to get much better efficiency than prior approaches.

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 [27] 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 [27] 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 [28, 9, 18, 14, 6]. 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 [22] further clarified the concept of Attribute-Based Encryption. In particular, they proposed two complimentary forms of ABE. In the first, Key-Policy ABE, attributes are used to annotate the ciphertexts and formula's over these

*Supported by NSF CNS-0524252, CNS-0716199, CNS-0749931; the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

attributes are ascribed to users' secret keys. The second type, Ciphertext-Policy ABE, is complimentary 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. [22] 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 [5]. 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 motivation for the problem of finding an expressive CP-ABE system under a more solid model. There have been multiple approaches in this direction.

First, in retrospect, we can view the basic Sahai-Waters[27] construction most"naturally" as Key-Policy ABE for a threshold gate. In their work, Sahai and Waters also describe how approach something akin to 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[17] 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 [21] provide a "bounded" CP-ABE construction. Their general approach was to show how to transform a KP-ABE system into a CP-ABE one. In particular, they provided a mapping onto a "universal" 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. The big drawback of their approach is that they need a *complete* tree of depth d to cover all possible access trees of depth d. Therefore, the size of this tree grows exponentially with d as $O(m^d)$, even though most used formulas of depth d will likely be of a size n that is much smaller. On the surface it appears that both the public parameter and private key will grow *exponentially* as $O(m^d)$. However, the authors point out that 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 $O(n^{3.42})$. Correspondingly, the important parameters of ciphertext and private key sizes add encryption and decryption complexity blow up by an $n^{3.42}$ factor. While this final transformation does provide a

[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.

polynomial solution, the blowup is too large to be of use in practice.

The authors describe their methods as being a generalization of the transformation onto Key Policy ABE techniques first used by Sahai and Waters. Indeed they make heavy use of dummy attributes methodology introduced by Sahai and Waters. The fact that these grafting methods lead to a large blowup in parameter and private key size, are arguably indicative that a "direct" approach is needed to achieve a system with reasonable efficiency.

Our Contribution We present new techniques for realizing Ciphertext-Policy ABE systems from a general set of access structures in the *standard model* under concrete and non-interactive assumptions. Our techniques allow for any attribute access structure that can be expressed by a Linear Secret Sharing Scheme (LSSS) matrix M. Previously used structures such as formulas (equivalently tree structures) can be expressed in terms of a LSSS. Thus our constructions permit slightly more expressive access structures than the Bethencourt, Sahai, and Waters construction.

Our techniques provide a framework for *directly* realizing CP-ABE systems. Within our framework we provide three different constructions each realizable under a different non-interactive assumption. In particular, we provide solutions under the decisional versions of the Bilinear Diffie-Hellman (BDH) assumption, the Bilinear-Diffie-Hellman Exponent (BDHE) assumption, and a new assumption we call the parallel Bilinear Diffie-Hellman assumption. These different constructions provide tradeoffs of the efficiency of the system versus the strength of the assumption used.

In all of our systems a ciphertext is associated with an $\ell \times n$ LSSS matrix M and function $\rho(\cdot)$ that maps rows of M to attribute strings. (One can think of each leaf node of a tree access structure scheme (e.g.,[22] corresponding to a row of M and ρ as a function describing the attribute of the leaf node). When an algorithm encrypts for a structure M it randomly shares a secret $s \in \mathbb{Z}_p$ according to the matrix M. A user's private key is associate 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.

The performance tradeoffs in our constructions reflect how efficiently 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.

We first show how to achieve a construction with much better efficiency and no setup limitation on the size of n under the ℓ^*-BDHE assumption. We refer to this as our "main" construction, since it will also serve as a template for our others. Our construction achieves much smaller ciphertexts of $O(\ell)$ group elements and private keys of size $O(|S|)$ group elements, matching the efficiency of the BSW construction. The main leverage point is that by using an assumption that gives the simulator ℓ^* different elements the simulator can compress a whole row of the challenge matrix into one group element of the parameters in the reduction.

The one drawback of this technique is that it can only work if an attribute appears at most once in a ciphertext. However, this can easily be overcome with a simple encoding of the attributes, where we 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.

Next, provide a construction overcomes this limitation by using an assumption that gives ℓ^* "parallel" instances of the BDHE problem. This extra information allows us to create a reduction

for a construction with $O(|S|)$ size private keys and no limitations on the amount of times an attribute is used.

Finally, we show how to modify the main scheme to realize a proof under the decisional Bilinear Diffie Hellman (BDH) assumption. In our BDH construction this results in ciphertexts consisting of $O(\ell \cdot n)$ group elements for a ciphertext $\ell \times n$ access matrix. A private key for an attribute set S consist of $O(|S| \times n_{\max})$ group elements, where $n_{\max}$ is the maximum width of any access matrix specified by the setup algorithm.

Taken all together our constructions provide a set of solutions to the Ciphertext-Policy ABE problem. Each solution is much more efficient than the Goyal, Jain, Pandey, and Sahai [21] construction. In addition, each of the assumption choices that we provide are much more preferable to the generic group model usedof Bethencourt, Sahai, and Waters [5] for proving security.

It is also interesting to consider the options provided with each assumption. Recently, there has been a large number of new number theoretic assumptions introduced in cryptography systems applying bilinear maps. Often, it is not clear where stronger assumptions are needed and where there are not. One contribution of this work is that it provides an exposition in this line of work of the tradeoffs one can make in the assumption complexity against a system's efficiency. On one hand, we show how an assumption that carries more information can be used to compress parameters and achieve more efficient systems. On the other, we show how to modify our system to obtain a weaker assumption.

1.1 Related Work

Some of the roots of ABE can be traced back to Identity-Based Encryption [28, 9, 18, 14, 6, 32, 19, 10] (IBE). One can view IBE as a very special case of ABE.

Different authors [30, 25, 3, 13] 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.

Since the introduction of Attribute-Based Encryption by Sahai and Waters [27], there have been several papers [22, 5, 16, 26, 21] 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 [26] 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 [31, 8, 1, 12, 11, 29, 24]. These systems have the advantages of hiding the associated access structures themselves and thus providing a level of "anonymity"; however, these systems 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 [23, 20] and Wildcard IBE [2].

1.2 Organization

In Section 2 we give relevant background on LSSS, the definitions for Ciphertext-Policy ABE, and our number theoretic assumptions. We begin by giving our decisional BDHE construction in Section 3. This construction will expose the general framework for our constructions and proofs. Next, we present a construction provably secure under the decisional parallel BDHE assumption in Section 4. Finally, we show how to realize CP-ABE under the decisional BDH assumption in Section 5 and conclude in Section 6.

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 [4]). *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., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$. 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 [4]:

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 [4] 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 [4] that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

2.3 Ciphertext-Policy ABE

An ciphertext-policy attribute based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt. In addition, we allow for the option of a fifth algorithm Delegate; however, in our exposition we don't explicitly give the delegation algorithm.

Setup. The setup algorithm takes no input other than the implicit security parameter. 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.

Delegate$(\mathrm{SK}, \tilde{S})$**.** The delegate algorithm takes as input a secret key SK for some set of attributes S and a set $\tilde{S} \subseteq S$. It output a secret key $\tilde{\mathrm{SK}}$ for the set of attributes $\tilde{S}$.

We now describe a security model for ciphertext-policy ABE schemes. Like identity-based encryption schemes [28, 9, 18] 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, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. 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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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 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 2.1. *We say that the decisional BDH assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional BDH problem.*

2.4.2 Decisional Bilinear Diffie-Hellman Exponent 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 2.2. *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.*

2.4.3 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 $\vec{y}$=

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

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

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

Definition 2.3. *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 Main Construction

We begin by giving our construction that is provably secure under the Bilinear Diffie-Hellman Exponent assumption. This construction is the simplest of the three and the proof illustrates how the assumption allows us to compress information into the parameters. We will later use it as a template for designing our other constructions. In the core construction an attribute can only be used in a most one row in the ciphertext access matrix M (the function $\rho(\cdot)$ is injective); however, this can be overcome with certain encodings of attributes. As illustrated in the introduction we can handle any particular attribute being used in up to $k_{\max}$ rows per ciphertext with a blowup of private keys by a factor of $k_{\max}$. For simplicity, we describe only the core construction without this transformation.

For ease of exposition, we describe our construction in the random oracle model; however, we show how to modify our scheme for a standard model solution in Appendix A. Our construction follows.

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 $H : \{0,1\}^* \to \mathbb{G}$ 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 $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^{n+1}$. 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

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

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 $\sum_{i \in I} \omega_i \lambda_i = s$. (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 $\mathcal{M}$.

3.1 Proof

We prove the following theorem.

Theorem 3.1. *Suppose the decisional q-BDHE assumption holds. Then no poly-time adversary can selectively break our system with a challenge matrix of size $\ell^* \times n^*$, 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 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, 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 responses from the oracle 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 $H(x)^t$ 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

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

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 $C = \mathcal{M}_\beta T$ 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

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

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.

3.2 Delegation, CCA-Security, and Removing Random Oracles

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 [15] by some simple alternations to the system that apply delegation. In the random oracle model standard methods for realizing CCA-security can be applied.

Finally, we can realize our construction in the standard model by applying a hash function similar to the one given by Sahai and Waters [27] and used in other subsequent work. In the setup suppose we restrict the access matrix to have at most $\ell_{\max}$ rows and users to have at most $Attr_{\max}$ attributes. Then we can replace the hash function $H(x)$ with a function $U(x) = g^{p(x)}$ for a random polynomial of degree $Attr_{\max} + \ell_{\max} - 1$. Details are given in Appendix A. We also note that the random oracle can be trivially removed if we restrict ourselves to a small fixed size of attributes in the system.

3.3 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

$$\rho(1) = \rho(2) = \text{“AttributeX}'' \quad \rho(3) = \rho(4) = \text{“AttributeY}''.$$

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.

4 An Unrestricted Construction

In the last section we described a construction that was naturally restricted to using an attribute at most once in the access structure. While we could work our way around this, using a naming transformation we need to pick a value $k_{\max}$ and restrict ourselves to having an attribute appear at most $k_{\max}$ times and having our private keys grow by a factor of $k_{\max}$.

We now show a new construction that is completely unrestricted and thus realizes efficiency on par with the Bethencourt, Sahai, and Waters [5] system, but under a concrete and non-interactive assumption. The main problem with our last system, as illustrated in the last subsection, is that the ciphertext components C_i can be combined. In this construction we make a small modification that further randomizes the ciphertexts to prevent against such an attack even when the ρ function is non-injective.

One big challenge is that the use of a non-injective ρ function complicates are proof methodology. In particular, in our reduction we programmed our random oracle for $H(x)$ based on the i-th row of M^* if $\rho^*(i) = x$. However, if there exist $i \neq j$ such that $x = H(i) = H(j)$ then there is an issue since we must program *both* row i and row j in the simulation. To deal with this we apply an assumption called 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.

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 $H : \{0,1\}^* \to \mathbb{G}$ 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 $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^{n+1}$. 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

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

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

4.1 Proof

We prove the following theorem.

Theorem 4.1. *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

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

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 $C = \mathcal{M}_\beta T$ 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 $\Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can play the decisional q-parallel BDHE game with non-negligible advantage.

5 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($n_{\max}$) 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 functions $H_1, \ldots, H_{n_{\max}}$ where $H_i : \{0,1\}^* \to \mathbb{G}$ that we will model as a random oracle. Note that it is simple to derive these n hash functions from one hash function. 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(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 $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^{n+1}$. These values will be used to share the encryption exponent s.

The ciphertext is published as

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

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

$$K = g^{\alpha} g^{at_1} \quad L_1 = g^{t_1}, \ldots, L_n = g^{t_{n_{\max}}} \quad \forall x \in S \quad K_x = \prod_{j=1,\ldots,n_{\max}} H_j(x)^{t_j}.$$

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

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

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

5.1 Proof

We prove the following theorem.

Theorem 5.1. *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 random oracles $H_1, \ldots, H_{n_{\max}}$ by building tables. Consider a call to $H_j(x)$. If $H_j(x)$ was already defined in the table, then simply return the same answer as before. Otherwise, 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

$$H_j(x) = g^{z_{x,j}} g^{a M_{i,j}}$$

Otherwise, let $H(x) = g^{z_{x,j}}$.

We point out a couple of facts. First, the responses from the oracle are 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 $\prod_j H(x)^j$ 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

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

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 $C = \mathcal{M}_\beta T$ and $C' = g^s$.

The tricky part is to simulate the $C_{i,j}$ values since the term $H_j(\rho^*(i))^s$ 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.

6 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.

Acknowledgements

We thank Matt Green for useful comments.

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

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

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

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

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

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

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

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

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

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

[13] 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.

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

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

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

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

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

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

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

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

[22] 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.

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

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

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

[26] 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.

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

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

[29] 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.

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

[31] 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.

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

A Removing the Random Oracle

In this section we describe how to realize our methodology in the standard model. In the proofs of our previous constructions 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.

We realize our standard model construction by adapting our main construction from Section 3. We then prove it secure under the decisional BDHE assumption in the standard model. We remark that similar techniques can be used to realize our other two constructions in the standard model, although we do not provide the details here. The adapted construction follows.

Setup($Attr_{\max}$,$\ell_{\max}$) 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 $m = Attr_{\max} + \ell_{\max} - 1$. 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

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

The authority sets $\text{MSK} = g^{\alpha}$ 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 $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^{n+1}$. 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 $\mathcal{M}$.

A.1 Proof

We prove the following theorem.

Theorem A.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 $n^* \leq q$ 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 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 all ℓ^* x values where we have 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 $x = \rho^*(i)$ we set $p_j(x) = M^*_{i,j}$ for $j \in [1, n^*]$.

We then conceptually will set

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

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 unambiguous.

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:

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

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:

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

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 $C = \mathcal{M}_\beta T$ 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

$$\vec{v} = (s, Sam + y_2', SSA^2 + y_3', \ldots, Sam^{n^*-1} + y_{n^*}') \in {\mathbb{Z}_p}^{n^*}.$$

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

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

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 n^*-BDHE game with non-negligible advantage.

B 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 [7].

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 $\vec{y}$=

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

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

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

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.