


Easy-ABE: An Easy Ciphertext-Policy Attribute-Based Encryption

Ahmad Koureich Ka 

Université Alioune Diop de Bambey, Senegal
ahmadkoureich.ka@uadb.edu.sn

Abstract. Attribute-Based Encryption is widely recognized as a leap forward in the field of public key encryption. It allows to enforce an access control on encrypted data. Decryption time in ABE schemes can be long depending on the number of attributes and pairing operations. This drawback hinders their adoption on a broader scale. In this paper, we propose a non-monotone CP-ABE scheme that has no restrictions on the size of attribute sets and policies, allows fast decryption and is adaptively secure under the CBDH-3 assumption. To achieve this, we approached the problem from a new angle, namely using a set membership relation for access structure. We have implemented our scheme using the Charm framework and the source code is available on GitHub. Easy-ABE performs better than FAME and FABEO.

Keywords: attribute-based encryption · CBDH-3 assumption · non-monotone · random oracle model.

1 Introduction

Traditionally, access control was applied to protect unencrypted data stored on servers. But the problem with this system is that if a server is compromised, the attacker gains direct access to unencrypted data. A solution to this problem appeared in 2005 on the proposal of Amit Sahai and Brent Waters [36] called attribute-based encryption. It allows to enforce fine-grained and flexible access controls over encrypted data.

Attribute-Based Encryption (ABE) is widely recognized as a leap forward in the field of public key encryption. It has numerous applications, ranging from cloud services [26], internet of things [5], video streaming [31], to healthcare systems [16].

ABE offers the possibility to encrypt for multiple recipients at once. Those who wanted to access the plaintext from the ciphertext simply had to have the necessary attributes to satisfy the ciphertext's built-in access control. ABE comes in two flavors: Key-Policy Attribute-Based Encryption (KP-ABE) in which the access policy is embedded in the recipients' secret keys and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) in which the access policy is embedded in the ciphertext. However, the design of such schemes faces many practical difficulties that hinder their wide adoption. A practical ABE scheme must have the following essential properties: [3,37]:

1. no restriction on the size of policies and attribute sets (unboundedness)
2. arbitrary string as an attribute (large universe);
3. based on the fast Type-3 pairings;
4. small number of pairings for decryption;
5. adaptive security under standard assumptions.

Many proposals have been made but few [3,37,33] satisfy the properties mentioned above. We believe that a practical ABE scheme of simple structure and more efficient can be constructed.

In this paper, we propose a non-monotonic ciphertext-policy attribute-based encryption denoted Easy-ABE. Our scheme not only offers constant-size secret keys, but also adds the above five properties. Unlike the majority of ABE schemes, Easy-ABE does not use a monotone span programs (MSP). An MSP consists of a matrix M and a function π that maps each row of M to an attribute. It is used as a linear secret sharing scheme. At a high level, Easy-ABE works as follow. Using the attribute universe, the user attribute set is mapped to a bit string called user attribute string. The encryption algorithm encrypts a plaintext under a set of authorized attribute strings. The users whose attribute strings appear on the set of authorized attribute strings gain access to the plaintext. We prove that the security of our CP-ABE scheme is based on the Computational Bilinear Diffie–Hellman assumption in Type-3 (CBDH-3).

Compared to FAME [3] and FABEO [33], the most efficient ABE schemes in the literature (to our knowledge), our scheme performs better. We have also implemented Easy-ABE using the Charm framework [4] and the source code is available on GitHub [23].

1.1 Related work

Attribute-Based Encryption is a natural extension of Identity Based Encryption [10,9]. It comes in two flavors: Key-Policy ABE and Ciphertext-Policy ABE. The first KP-ABE scheme was presented by Goyal et al. [20]. Ciphertext-Policy ABE was first proposed by Bethencourt et al. [8] who prove its security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. It is followed by the work of Cheung et al. [14] in which access structures are AND gates on positive and negative attributes. They improve the security of their scheme by applying the Canetti-Halevi-Katz technique to obtain a chosen ciphertext attack (CCA) security.

Other attribute-based encryption schemes have emerged focusing on constant-size ciphertexts [6,13,40,17,38,15,22,39]. In [6], the private key size is linear in the number of attributes of the user. To address the efficiency problem that plagues many schemes due to high computational cost [38] also provides constant computational cost useful when computational and bandwidth issues are major concerns. [28,21,24] have designed constant-size secret key schemes. Besides the constant-size secret key, [28] provides low computational and storage overhead with an expressive AND-gate access structure.

Some schemes take scalability into account. They are called unbounded schemes. Unboundedness is an essential property for an ABE scheme because it allows adding new attributes without having to redeploy the scheme. Lewko and Waters [27] were the first to present such a scheme followed by [29,12,35,16,3,37]. In [16] unboundedness is obtained by limiting the attribute elements in the ciphertexts to only those associated with the attribute group keys of the ciphertext attributes.

Most of the schemes in the literature are monotonous since it is natural to admit that a user having more attributes than required to access an information must have access to that information. But this could give rise to conflict of interest. Non-monotonic ABE schemes have also been proposed [37,30,29,16]. But some of these proposals [30,29] are inefficient when it comes to decryption and storage.

Many ABE schemes [14,16,17,20,21,28,30,35,38,40] offering various attractive properties (such as constant-size ciphertexts, constant-size secret key, scalability, unboundedness) have been shown to be secure only in the selective security model which is weaker than the adaptive security enjoyed by our scheme. The problem with selective security is that for the deployment of the scheme one adversary have to declare the access structure he wants to attack. Which is very unlikely to happen in reality.

The first KP-ABE and CP-ABE schemes that exhibit the five essential properties (listed above) are proposed by Agrawal and Chase [3]. They named their proposals FAME. It is in their wake that Tomida and Kawahara proposed KP-ABE and CP-ABE schemes [37] that additionally allow the use of negation and multi-use of attributes in policy. Recently, in this category of practical schemes, Riepel and Wee proposed a KP-ABE and CP-ABE schemes named FABEO [33]. FABEO features improved efficiency and quantitatively stronger safety guarantee. It exceeds FAME on all parameters of practical interest (ciphertext size, key sizes and running time).

1.2 Organization

This paper is structured as follows: after the introduction, section 2 presents the notations, terminologies and tools necessary for the formal description in section 3 of our CP-ABE scheme called Easy-ABE. The proof of security of our scheme is done in section 4. In sections 5 and 6, we present a theoretical comparison and a performance comparison of Easy-ABE with FAME and FABEO. Section 7 concludes this work.

2 Preliminaries

In this section, we present the notations, terminologies and tools necessary for the presentation of Easy-ABE.

2.1 Access structures

We denote by $\mathcal{U} = \{A_1, A_2, \dots, A_l\}$ the universe of attributes where $A_{i,i=1,\dots,l}$ are attributes. In our scheme, a set of user attributes $\mathcal{S} \subseteq \mathcal{U}$ is mapped to an $|\mathcal{U}|$ -bit string $\omega = b_l \dots b_1$ where $l = |\mathcal{U}|$ and

$$b_i = \begin{cases} 0 & \text{if } A_i \notin \mathcal{S} \\ 1 & \text{otherwise} \end{cases}$$

In the rest of the paper, we call ω a user attribute string. For example, if $\mathcal{S} = \{A_2, A_4, A_5\}$, the user attribute string ω will be $0 \dots 011010 \in \{0, 1\}^l$.

To generate the user's secret key, ω will be taken as a binary number in \mathbb{Z}_p^* and then mapped to a group element. Representing ω in reverse order (of the indexes) makes our schema scalable since by expanding the universe of attributes (addition of new attributes) the keys generated before the expansion remain valid and no re-encryption of data is needed. For example, if \mathcal{U} expands to $\{A_1, A_2, \dots, A_l, \dots, A_n\}$, the user attribute set $\mathcal{S} = \{A_2, A_4, A_5\}$ will be represented by $0 \dots 011010 \in \{0, 1\}^n$ and will remain unchanged when considered as a binary number. This representation of the set of user attributes guarantees unboundedness and large universe (since any arbitrary string can be used as an attribute) to our scheme.

Definition 1. (Access Structure). *We say that an access structure $\mathbb{A} \subseteq \{0, 1\}^n$ is the set of authorized user attribute strings. That is a user attribute string ω is authorized if and only if $\omega \in \mathbb{A}$.*

From our definition of the access structure, it is clear that our scheme is non-monotonic since a monotonic access structure is defined as follows:

Definition 2. (Monotonic Access Structure [3]). *If \mathcal{U} denotes the universe of attributes, then an access structure \mathbb{A} is a collection of non-empty subsets of \mathcal{U} , i.e., $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \emptyset$. It is called monotone if for every $B, C \subseteq \mathcal{U}$ such that $B \subseteq C, B \in \mathbb{A} \Rightarrow C \in \mathbb{A}$.*

Although it is non-monotonic, Easy-ABE becomes monotonic if it is accepted for a user to query for a secret key associated to a subset of her/his set of attributes.

The access structure can also be defined without the use of a universe of attributes by considering directly the set of binary strings representing authorized users identities. For example, informations on citizen id card can be hashed to serve as user attribute string.

Multi-use of attributes. ABE schemes that rely on MSPs are restricted to accept 0 or 1 occurrences of attributes in access policy. This is called the one-use restriction. Therefore, the access policy (`Degree:Doctoral AND AgeGroup:<30`) OR (`Degree:Master AND Field:Engineering`) OR (`Degree:Doctoral AND Field:Education`) is not accepted because `Degree:Doctoral` is repeated. The one-use restriction comes from the fact that the mapping function π in an MSP (M, π) must be injective, i.e., distinct rows in M should be mapped to distinct attributes.

These schemes may lift the single-use restriction at the cost of a larger key in CP-ABE schemes or a larger ciphertext in KP-ABE schemes, thus leading to a loss of efficiency.

Easy-ABE does not suffer of this practical drawback and natively supports the multi-use of attributes. Given the universe of attributes $\{\text{Degree:Master}, \text{Degree:Doctoral}, \text{Field:Education}, \text{Field:Engineering}, \text{AgeGroup}<30\}$, our scheme converts the above access policy into the set of authorized attribute strings $\{10010, 01001, 00110\}$. Easy-ABE has no restrictions on policies.

2.2 Ciphertext-Policy ABE

A Ciphertext-Policy Attribute-Based Encryption consists of four algorithms (adapted from [8]):

- **Setup**(λ): The algorithm takes a security parameter λ and outputs the system parameters params , a master public key mpk and a master secret key msk .
- **Encrypt**($\text{mpk}, \mathbb{A}, m$): The algorithm takes the master public key mpk , a set of authorized user attribute strings \mathbb{A} and a message m then outputs a ciphertext ct .
- **KeyGen**($\text{mpk}, \text{msk}, \omega$): The algorithm takes the master public key mpk , the master secret key msk and a user attribute string ω then outputs a secret key sk .
- **Decrypt**(ct, sk): The algorithm takes a ciphertext ct and a secret key sk then outputs the plaintext m or \perp .

2.3 Security model

In this paper, we are interested in indistinguishability under chosen plaintext attack (IND-CPA) modelled by the following IND-CPA game between a challenger \mathcal{C} and an adversary \mathcal{A} :

- **Setup**. \mathcal{C} runs the Setup algorithm of a CP-ABE scheme denoted Π with the security parameter λ as input and gives params and mpk to \mathcal{A} .
- **Phase 1**. \mathcal{A} can make repeated (at will) secret key queries for user attribute strings ω and receives from \mathcal{C} their corresponding secret keys.
- **Challenge**. \mathcal{A} submits a set of user attribute strings \mathbb{A} and two messages m_0, m_1 of the same length. One restriction should be noted: the intersection of \mathbb{A} and the set of user attribute strings used for secret key queries must be empty. \mathcal{C} randomly selects $b \in \{0, 1\}$, encrypts the message m_b and sends the result to the \mathcal{A} .
- **Phase 2**. Similar to phase 1 with the restriction that the intersection of \mathbb{A} and the set of user attribute strings used for secret key queries must be empty.
- **Guess**. The adversary outputs a guess b' of b . We say that \mathcal{A} succeeded if $b' = b$.

The advantage of an adversary \mathcal{A} in the IND-CPA game is defined as

$$\text{Adv}_{\Pi}^{\mathcal{A}}(\lambda) = \Pr[b' = b] - \frac{1}{2}$$

Definition 3. A CP-ABE scheme Π is fully or adaptively IND-CPA secure if for any polynomial time adversary \mathcal{A} , $\text{Adv}_{\Pi}^{\mathcal{A}}$ is negligible, that is to say $\text{Adv}_{\Pi}^{\mathcal{A}}$ is smaller than the inverse of any polynomial, for all large enough values of λ .

2.4 Bilinear maps and Diffie–Hellman assumption

Let \mathbb{G}_1 , \mathbb{G}_2 and \mathbb{G}_T be three cyclic groups of prime order p . A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following properties [18,11]:

1. Bilinearity: $e(g_1g'_1, g_2g'_2) = e(g_1, g_2)e(g_1, g'_2)e(g'_1, g_2)e(g'_1, g'_2)$ for all $g_1, g'_1 \in \mathbb{G}_1, g_2, g'_2 \in \mathbb{G}_2$.
2. Non-degeneracy: for any $g_1 \in \mathbb{G}_1$, if $e(g_1, g_2) = 1$ for all $g_2 \in \mathbb{G}_2$, then $g_1 = 1$ (and similarly with $\mathbb{G}_1, \mathbb{G}_2$ reversed).
3. Computability: The map e is efficiently computable.

The pairing is asymmetric when $\mathbb{G}_1 \neq \mathbb{G}_2$ and of Type-3 when no efficiently-computable isomorphism is known from \mathbb{G}_2 to \mathbb{G}_1 (or from \mathbb{G}_1 to \mathbb{G}_2).

Definition 4. (Bilinear group generator [18]). For the purpose of simplicity we say that an asymmetric bilinear group generator is an algorithm \mathcal{G} that takes as input a security parameter λ and outputs a description of three groups $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T of prime order p . We assume that this description permits efficient (polynomial-time in λ) group operations and random sampling in each group. The algorithm also outputs an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and generators g_1 and g_2 for \mathbb{G}_1 and \mathbb{G}_2 , respectively.

Definition 5. (Computational Bilinear Diffie–Hellman Problem in Type-3 (CBDH-3) [11]). Given $g_1^\alpha, g_1^\beta, g_1^\gamma \in \mathbb{G}_1$ and $g_2^\beta, g_2^\gamma \in \mathbb{G}_2$ for $\alpha, \beta, \gamma \in_{\mathbb{R}} \mathbb{Z}_p^*$, the CBDH-3 problem is to compute the Type-3 pairing value $e(g_1, g_2)^{\alpha\beta\gamma}$. The CBDH-3 assumption asserts that CBDH-3 problem is hard. That is to say for all PPT adversaries \mathcal{A} the advantage:

$$\text{Adv}_{\text{CBDH-3}}^{\mathcal{A}}(\lambda) = \Pr \left[\mathcal{A} \left(\begin{array}{c} \text{pair-grp}, \\ g_1^\alpha, g_1^\beta, g_1^\gamma \in \mathbb{G}_1, \\ g_2^\beta, g_2^\gamma \in \mathbb{G}_2 \end{array} \right) = e(g_1, g_2)^{\alpha\beta\gamma} \mid \begin{array}{c} \text{pair-grp} \leftarrow \mathcal{G}(\lambda), \\ \alpha, \beta, \gamma \in_{\mathbb{R}} \mathbb{Z}_p^* \end{array} \right]$$

is negligible in λ , where $\text{pair-grp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$.

2.5 Some cryptographic primitives

In this section, we briefly present some cryptographic primitives used in our scheme.

Symmetric encryption scheme. A symmetric encryption scheme (SYM) is a pair of probabilistic polynomial-time algorithms (Enc , Dec) such that:

1. The encryption algorithm Enc takes as input a key $k \in \{0, 1\}^n$ (n is related to a security parameter) and a plaintext message $m \in \{0, 1\}^*$, and returns a ciphertext $c \in \{0, 1\}^*$.
2. The decryption algorithm Dec takes as input a key k and a ciphertext c , and returns a message m .

It is required that for all k and m , $\text{Dec}_k(\text{Enc}_k(m)) = m$.

The symmetric encryption scheme is said to be secure in the sense of INDistinguishability under Chosen-Plaintext Attacks (IND-CPA) if from the encryption of one of its two messages, the adversary cannot tell which one has been encrypted even if it has knowledge of encryptions of many other messages of its choice.

Message authentication codes. A message authentication code (MAC) is a pair of probabilistic polynomial-time algorithms (Mac , Vrfy) such that:

1. The tag-generation algorithm Mac takes as input a key $k \in \{0, 1\}^n$ (n is related to a security parameter) and a message $m \in \{0, 1\}^*$, and returns a tag t .
2. The verification algorithm Vrfy takes as input a key k , a message m and a tag t . It returns 1 indicating that t is valid, thus m is authentic and 0 indicating that t is invalid, thus m is unauthentic.

It is required that for all k and m , $\text{Vrfy}_k(m, \text{Mac}_k(m)) = 1$.

The message authentication code is said to satisfy Strong Unforgeability under Chosen-Message Attacks (SUF-CMA) it is computationally infeasible for the adversary to provide a new tag t for a message m even if it has knowledge of many other tags for messages of its choice [7].

Key derivation function. Formally, a key derivation function (KDF) is defined in [25] as an algorithm that takes as input four arguments: a value σ sampled from a source of keying material, a value l indicating the length of the secret key to return, and two additional arguments, a salt value r defined over a set of possible salt values and a context variable c , both of which are optional, i.e., can be set to the null string or to a constant.

Informally, the key derivation function is said to be secure if its output distribution is computationally indistinguishable from the uniform distribution over $\{0, 1\}^l$.

In our scheme, the source of keying material is an algebraic group.

2.6 Diffie-Hellman Integrated Encryption Scheme

Abdalla et al [1] suggested a method for encrypting strings using the Diffie-Hellman assumption. The method is called Diffie-Hellman Integrated Encryption

Scheme (DHIES) and is secure against chosen-ciphertext attack. The version [19] we describe here uses a symmetric encryption scheme $\text{SYM} = (\text{Enc}, \text{Dec})$ a message authentication code $\text{MAC} = (\text{Mac}, \text{Vrfy})$ and a key derivation function KDF.

Let \mathbb{G} be a finite cyclic group of order p generated by g . Let $a \in_R \mathbb{Z}_p^*$ and $h = g^a$. The public key is (\mathbb{G}, g, h) and the private key is a .

Encrypt(\mathbf{m}, h): To encrypt $\mathbf{m} \in \{0, 1\}^*$, do the following:

1. Choose a random $k \in \mathbb{Z}_p^*$ and set $c_1 = g^k$
2. Set $\kappa = \text{KDF}(h^k, l_1 + l_2)$ and parse κ as $\kappa_1 || \kappa_2$ where κ_1 and κ_2 are l_1 and l_2 bit binary strings respectively.
3. Set $c_2 = \text{Enc}_{\kappa_1}(\mathbf{m})$ and $c_3 = \text{Mac}_{\kappa_2}(c_2)$.
4. Transmit the ciphertext (c_1, c_2, c_3) .

Decrypt(c_1, c_2, c_3, a):

1. Compute $\kappa = \text{KDF}(c_1^a, l_1 + l_2)$ and parse it as $\kappa_1 || \kappa_2$ where κ_1 and κ_2 are l_1 and l_2 bit binary strings respectively.
2. Check whether $\text{Vrfy}_{\kappa_2}(c_3, \text{Mac}_{\kappa_2}(c_2)) = 1$ (if not then return \perp and halt).
3. Return $\mathbf{m} = \text{Dec}_{\kappa_1}(c_2)$.

3 Easy-ABE: Our CP-ABE scheme

In this section, we give a formal description of the four algorithms (Setup, Encrypt, KeyGen and Decrypt) that characterise Easy-ABE:

Setup(λ): To produce the system parameters params , the master public key mpk and the master secret key msk , the algorithm performs the following steps:

- (1) Run $\mathcal{G}(\lambda)$ to obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$.
- (2) Choose two cryptographic hash functions:

$$H_1 : \{0, 1\}^n \rightarrow \mathbb{G}_1 \quad \text{and} \quad H_2 : \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$$

- (3) Choose two random exponents α and $\beta \in \mathbb{Z}_p^*$.
- (4) Since our scheme uses a Diffie-Hellman Integrated Encryption Scheme (DHIES) [1], choose an IND-CPA secure symmetric encryption scheme $\text{SYM} = (\text{Enc}, \text{Dec})$ a SUF-CMA secure message authentication code $\text{MAC} = (\text{Mac}, \text{Vrfy})$ and a secure key derivation function KDF.
- (5) Return the system parameters, the master public key and the master secret key:

$$\text{params} = (H_1, H_2, \text{SYM}, \text{MAC}, \text{KDF})$$

$$\text{mpk} = (g_1, g_1^\alpha, g_1^\beta, g_2, g_2^\beta)$$

$$\text{msk} = (\alpha, \beta)$$

KeyGen(mpk, msk, ω): To produce the user secret key for the user attribute string ω , the algorithm performs the following steps:

- (1) Compute $h_\omega = H_1(\omega) \in \mathbb{G}_1$.
- (2) Pick a random $r \in \mathbb{Z}_p^*$ and return the secret key

$$\text{sk} = (g_1^{\alpha\beta} h_\omega^r, g_2^r)$$

Encrypt(mpk, \mathbb{A} , \mathbf{m}): To encrypt a message $\mathbf{m} \in \{0, 1\}^*$ under the set of authorized user attribute strings \mathbb{A} , the algorithm performs the following steps:

- (1) Pick a random exponent $s \in \mathbb{Z}_p^*$.
- (2) Compute $h_\omega = H_1(\omega) \in \mathbb{G}_1$, $\forall \omega \in \mathbb{A}$.
- (3) Compute $\sigma = H_2(e(g_1^\alpha, g_2^\beta)^s)$.
- (4) Perform a DHIES [1].
 - (a) Pick a random $k \in \mathbb{Z}_p^*$, set $c_1 = g_1^\beta g_1^k$.
 - (b) Compute $\kappa = \text{KDF}(c_1^\sigma, l_1 + l_2)$ and split the binary string κ in two substrings κ_1 and κ_2 of length l_1 and l_2 respectively ($\kappa = \kappa_1 || \kappa_2$). l_1 and l_2 must match the key lengths of the underlying SYM and MAC schemes respectively.
 - (c) Compute $c_2 = \text{Enc}_{\kappa_1}(m)$ and $c_3 = \text{Mac}_{\kappa_2}(c_2)$.
- (5) Return the ciphertext:

$$\text{ct} = (c_1, c_2, c_3, g_2^s, \{h_\omega^s\}_{\omega \in \mathbb{A}})$$

Decrypt(ct, sk): To decrypt the ciphertext ct with the user secret key sk, the algorithm performs the following steps:

- (1) Find h_ω^s in ct with the same index ω as in $g_1^{\alpha\beta} h_\omega^r$ in sk. If the search fails then return \perp and halt.
- (2) Otherwise, let h_ω^s be the returned value from the search then compute:

$$\begin{aligned} \rho &= e(g_1^{\alpha\beta} h_\omega^r, g_2^s) / e(h_\omega^s, g_2^r) \\ &= e(g_1, g_2)^{\alpha\beta s} e(h_\omega, g_2)^{rs} / e(h_\omega, g_2)^{sr} \\ &= e(g_1, g_2)^{\alpha\beta s} \end{aligned} \tag{1}$$

- (3) Compute $\sigma = H_2(\rho)$.
- (4) Recover m with DHIES [1].
 - (a) Compute $\kappa = \text{KDF}(c_1^\sigma, l_1 + l_2)$ and split the binary string κ in two substrings κ_1 and κ_2 of length l_1 and l_2 respectively ($\kappa = \kappa_1 || \kappa_2$). l_1 and l_2 must match the key lengths of the underlying SYM and MAC schemes respectively.
 - (b) Compute $b = \text{Vrfy}_{\kappa_2}(c_3, \text{Mac}_{\kappa_2}(c_2))$, if $b = 0$ then return \perp and halt.
 - (c) Otherwise, compute $m = \text{Dec}_{\kappa_1}(c_2)$.
- (5) Return the plaintext m .

4 Security analysis

The security of Easy-ABE is based on the hardness of the Computational Bilinear Diffie–Hellman Problem in Type-3 (CBDH-3). Assuming that the underlying primitives (SYM, MAC and KDF) are secure, we show that the proposed scheme has ciphertext indistinguishability against chosen plaintext attack (IND-CPA) in the random oracle model.

Theorem 1. *Let H_1 and H_2 be two random oracles. Let SYM be a symmetric encryption scheme, let MAC be a message authentication code and let KDF be a key derivation function. If there exists an IND-CPA adversary \mathcal{A} that has advantage $\epsilon(\lambda)$ against Easy-ABE by making q secret key queries and a challenge set of size m of user attribute strings, then there exists a simulator that solves the CBDH-3 problem with advantage at least $m\epsilon(\lambda)/q$ and a running time $\mathcal{O}(\text{time}(\mathcal{A}))$.*

Proof. The simulator is given a random instance of the CBDH-3 problem: $g_1, g_1^a, g_1^b, g_1^c \in \mathbb{G}_1$ and $g_2, g_2^b, g_2^c \in \mathbb{G}_2$ for a, b and c randomly chosen from \mathbb{Z}_p^* . The simulator must output the solution of the CBDH-3 problem that is $e(g_1, g_2)^{abc} \in \mathbb{G}_T$. The groups $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are of prime order p , g_1 and g_2 are generators of \mathbb{G}_1 and \mathbb{G}_2 respectively.

Now we describe how the simulator uses adversary \mathcal{A} to solve the CBDH-3 problem in the following IND-CPA security game:

Setup: The simulator runs the Setup algorithm of Easy-ABE and sends to \mathcal{A} the system parameters and the master public key:

$$\text{params} = (H_1, H_2, \text{SYM}, \text{MAC}, \text{KDF})$$

$$\text{mpk} = (g_1, g_1^a, g_1^b, g_2, g_2^b)$$

Note that the two hash functions H_1 and H_2 are random oracles controlled by the simulator as described below.

H_1 -queries: The adversary \mathcal{A} can send an H_1 query at any time. Therefore the simulator maintains a list initially empty of the form $(\omega, t, h_\omega) \in \{0, 1\}^n \times \mathbb{Z}_p^* \times \mathbb{G}_1$. This list is denoted H_1 -list. When a query $H_1(\omega)$ is received, the simulator checks if the tuple (ω, t, h_ω) is in the list. If yes responds with $H_1(\omega) = h_\omega$. Otherwise it picks a random $t \in \mathbb{Z}_p^*$, computes $h_\omega = g_1^a g_1^t \in \mathbb{G}_1$, adds the tuple (ω, t, h_ω) to the list and responds with $H_1(\omega) = h_\omega$.

H_2 -queries: The adversary \mathcal{A} can send an H_2 query at any time. Therefore the simulator maintains a list initially empty of the form $(x, y) \in \mathbb{G}_T \times \mathbb{Z}_p^*$. This list is denoted H_2 -list. When a query $H_2(x)$ is received, the simulator checks if the tuple (x, y) is in the list. If yes responds with $H_2(x) = y$. Otherwise, it picks a random $y \in \mathbb{Z}_p^*$, adds the tuple (x, y) to the list and responds with $H_2(x) = y$.

Phase 1: The adversary \mathcal{A} can send secret key queries for user attribute strings $\{\omega_i\}_{i=1 \dots l}$ of its choice. Therefore, the simulator maintains a list initially empty of the form $(\omega_i, \text{sk}_i = (\mu_i, \nu_i))$. This list is denoted **sk-list**. When a secret key query for $\omega_i \in \{0, 1\}^n$ is received, the simulator runs the H_1 -**queries** algorithm to obtain the tuple $(\omega_i, t_i, h_{\omega_i})$ corresponding to ω_i from the H_1 -list, picks a random $r \in \mathbb{Z}_p^*$ and responds to \mathcal{A} with

$$\text{sk}_i = \left((g_1^b)^{-t_i} h_{\omega_i}^r, g_2^r / g_2^b \right) \quad (2)$$

The simulator adds the tuple (ω_i, sk_i) to the **sk-list**. One can see that sk_i is a valid secret key because

$$(g_1^b)^{-t_i} h_{\omega_i}^r = g_1^{ab} g_1^{-ab} g_1^{-bt_i} (g_1^a t_i)^r = g_1^{ab} (g_1^a t_i)^{(r-b)} = g_1^{ab} h_{\omega_i}^{(r-b)} = g_1^{ab} h_{\omega_i}^{\tilde{r}}$$

which shows that (2) satisfies $(g_1^{ab} h_{\omega_i}^{\tilde{r}}, g_2^{\tilde{r}})$ matching the definition of a secret key in the **KeyGen** algorithm of Easy-ABE.

Challenge: When the adversary feels ready for the challenge, it submits a set of user attribute strings \mathbb{A} and two messages $m_0, m_1 \in \{0, 1\}^*$ of the same length. If there is an ω present at the same time in both \mathbb{A} and **sk-list** then the simulator terminates its interactions with \mathcal{A} and outputs abort. Otherwise the simulator selects $m_{b, b \in \{0, 1\}}$, picks two random $c_2, c_3 \in \{0, 1\}^*$ (with the appropriate lengths related to the output-length of **Enc** and **Mac**), m random $s_i \in \mathbb{Z}_p^*$ where $m = |\mathbb{A}|$ the number of elements in \mathbb{A} . The simulator responds to \mathcal{A} with the ciphertext

$$\text{ct} = (g_1^c, c_2, c_3, g_2^c, \{(g_1^c)^{s_i}\}_{i=1 \dots m}) = (g_1^c, c_2, c_3, g_2^c, \{(g_1^{s_i})^c\}_{i=1 \dots m})$$

Note that the tuple (g_1^c, c_2, c_3) is indistinguishable from the encryption of a random message m as in the real ciphertext since the underlying primitives (**SYM**, **MAC** and **KDF**) are secure [34]. Also, note that from the H_1 -**queries** algorithm each $\omega \in \mathbb{A}$ is mapped to a random element $h_\omega \in \mathbb{G}_1$ as the $g_1^{s_i}$ for $i = 1 \dots m$ are random in \mathbb{G}_1 . Therefore, **ct** is a valid ciphertext.

One can see that with a secret key of the form $\text{sk}_i = (g_1^{ab} g_1^{s_i r}, g_2^r)$ for $i \in [1 \dots m]$ the solution $e(g_1, g_2)^{abc}$ of the instance of the CBDH-3 problem given to the simulator appears in the **Challenge** phase.

Phase 2: \mathcal{A} sends more secret key queries for user attribute strings $\{\omega_i\}_{i=l+1 \dots q}$ of its choice as in **Phase 1** with the restriction that none of the ω_i is in the set of user attribute strings \mathbb{A} used in the **Challenge** phase.

Guess: Adversary \mathcal{A} outputs its guess $b' \in \{0, 1\}$. The simulator selects a random tuple $(\omega_j, \text{sk}_j = (\mu_j, \nu_j))$ from the **sk-list**, computes $\rho = e(\mu_j, g_2^c) / e(g_1^{s_i c}, \nu_j)$ for an $i \in [1 \dots m]$ and outputs ρ as the solution to the given instance of CBDH-3 problem.

The fact that the secret keys and the ciphertext sent to \mathcal{A} from queries are valid justifies that the view of \mathcal{A} when used by the simulator is distributed identically to \mathcal{A} 's view in a real attack against Easy-ABE.

The simulator outputs the correct solution when $b' = b$ and there is a tuple in the H_1 -list of the form $(*, *, g^{s_i})_{i \in [1 .. m]}$ produced when the **Phase 1** algorithm is run. Let H_1 -sublist be the subset of H_1 -list containing tuples produced by executions of the **Phase 1** algorithm and let \mathcal{E} be the event that the simulator's output is correct. Let \mathcal{B} be the event that there exist an $i \in [1 .. m]$ such that the tuple $(*, *, g^{s_i})$ appears in H_1 -sublist. We have:

$$\begin{aligned} \Pr[b' = b] &= \Pr[b' = b \mid \mathcal{B}] \cdot \Pr[\mathcal{B}] + \Pr[b' = b \mid \overline{\mathcal{B}}] \cdot \Pr[\overline{\mathcal{B}}] \\ &\leq \Pr[b' = b \mid \mathcal{B}] + \Pr[b' = b \mid \overline{\mathcal{B}}] \cdot \Pr[\overline{\mathcal{B}}] \end{aligned} \quad (3)$$

If event \mathcal{B} does not occur, then the view of \mathcal{A} in its interactions with the simulator is independent of $e(g_1, g_2)^{abc}$. Therefore $\Pr[b' = b \mid \overline{\mathcal{B}}] = 1/2$ and (3) becomes:

$$\Pr[b' = b] \leq \Pr[b' = b \mid \mathcal{B}] + \frac{1}{2} \Pr[\overline{\mathcal{B}}] \leq \Pr[b' = b \mid \mathcal{B}] + \frac{1}{2} \quad (4)$$

which leads to

$$\Pr[b' = b \mid \mathcal{B}] \geq \Pr[b' = b] - \frac{1}{2} = \text{Adv}_{\Pi}^{\mathcal{A}}(\lambda) = \epsilon \quad (5)$$

To complete the proof of the theorem, we have:

$$\begin{aligned} \Pr[\mathcal{E}] &= \Pr[b' = b \wedge \mathcal{B}] \\ &= \Pr[b' = b \mid \mathcal{B}] \cdot \Pr[\mathcal{B}] \\ &= \Pr[b' = b \mid \mathcal{B}] \cdot \Pr[\exists i \text{ s.t. } (*, *, g^{s_i}) \in H_1\text{-sublist}] \\ &\geq \epsilon \cdot \sum_{i=1}^m \Pr[(*, *, g^{s_i}) \in H_1\text{-sublist}] = m\epsilon/q \end{aligned}$$

□

5 Theoretical Comparison

We compare Easy-ABE with FAME and FABEO. In this comparison, we consider the one-use restricted versions of FAME and FABEO. We are interested in two metrics that are the computational cost and the storage cost. For the computational cost, we compare the number of multiplications, exponentiations, hash function calls and pairings in key-generation, encryption and decryption algorithms, see tables 1 to 3. For the storage cost we compare the number of group elements in secret keys and ciphertexts, see table 4.

Table 1. Comparison of the number of operations for Key generation. T denotes the number of attributes input to the KeyGen algorithm.

Scheme	Key generation					
	\mathbb{G}_1			\mathbb{G}_2		
	Mul	Exp	Hash	Mul	Exp	Hash
FAME	$8T + 9$	$9T + 9$	$6(T + 1)$	–	3	–
FABEO	1	$T + 2$	$T + 1$	–	1	–
Our	1	2	1	–	1	–

Table 2. Comparison of the number of operations for encryption. m is the number of attribute strings in our access policy \mathbb{A} and n_1, n_2 are the number of rows and columns of the MSP in FAME and FABEO.

Scheme	Encryption					
	\mathbb{G}_1			\mathbb{G}_2		
	Mul	Exp	Hash	Mul	Exp	Hash
FAME	$12n_1n_2 + 6n_1$	$6n_1$	$6(n_1 + n_2)$	–	3	–
FABEO	n_1	$2n_1$	$n_1 + 1$	–	2	–
Our	1	$m + 2$	m	–	1	–

Table 3. Comparison of the number of operations for decryption. I is the number of attributes used in decryption.

Scheme	Decryption			
	Multiplication			Pairing
	\mathbb{G}_1	\mathbb{G}_2	\mathbb{G}_T	
FAME	$6I + 3$	–	6	6
FABEO	$2I$	–	3	3
Our	–	–	1	2

Table 4. Comparison of the size of keys and ciphertexts. m is the number of attribute strings in our access policy \mathbb{A} , T denotes the number of attributes input to the KeyGen algorithm and n_1 is the number of rows of the MSP in FAME and FABEO.

Scheme	Storage cost			
	Key size		Ciphertext size	
	\mathbb{G}_1	\mathbb{G}_2	\mathbb{G}_1	\mathbb{G}_2
FAME	$3(T + 1)$	3	$3n_1$	3
FABEO	$T + 1$	1	n_1	2
Our	1	1	$m + 1$	1

From these tables, it is clear that our scheme is more efficient than FAME and FABEO in terms of computational cost and storage cost. This performance mainly comes from mapping the user attribute set into a bit string and converting the access policy into a set of authorized attribute strings. It should be noted that in the comparison we did not take into account the running time of the symmetric encryption scheme, the running time of the message authentication

code and the complexity of the search algorithm used in the first step of our decryption algorithm. The following section will shed some light on this issue.

6 Implementation

We implemented Easy-ABE in Python 3.7.17 using the Charm framework version 0.50. The Type-3 MNT224 curve is used for pairings. We used the implementations of FAME [2] and FABEO [32] for performance comparisons. The experiments are carried out on an Ubuntu 22.04 desktop computer equipped with an Intel Core i5-4590S CPU @ 3.00GHz \times 4 and 8GB RAM.

It is worth noting that for the performance evaluation we use FAME and FABEO as ABE key encapsulation mechanism (ABE-KEM). Therefore, we extend their encryption and decryption algorithms so that they can encrypt large messages as Easy-ABE.

We performed 2 sets of experiments. In the first one we compared the running times for the setup, key generation, encryption and decryption algorithms with AND policies "1 AND 2 AND ... AND N " for $N \in \{10, 20, \dots, 100\}$ as in [3] and [33]. In this case all the N attributes are required for decryption. In the second one we compared the running times of the encrypt and decrypt algorithms with OR policies "1 OR 2 OR ... OR N " for $N \in \{10, 20, \dots, 100\}$. In this case only one attribute is required for decryption. This is motivated by the fact that Easy-ABE converts an AND policy of size N into a singleton set of authorized attribute strings while the OR policy of size N is converted into a set of N authorized attribute strings.

For each experiment, we use the one-use restricted versions of FAME and FABEO. The average running time of each algorithm is computed from 20 executions as in [33].

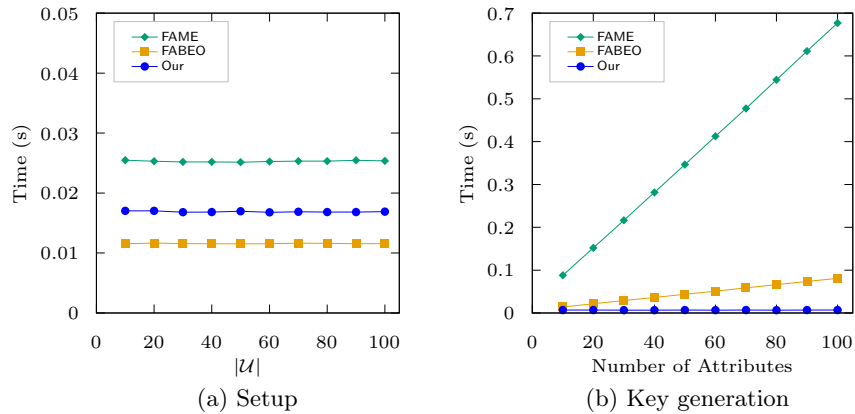


Fig. 1. Average running times for the setup and key generation algorithms with AND policies. $|\mathcal{U}|$ is the size of the universe of attributes.

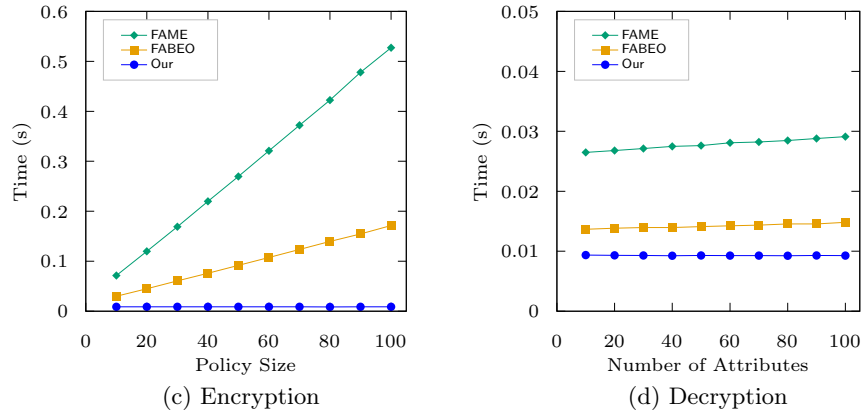


Fig. 2. Average running times for the encryption and decryption algorithms with AND policies. For the encryption we use a random 16 bytes plaintext.

Figure 1 and 2. Not surprisingly, the setup times are independent of the size of the universe of attributes. However, figure 1(a) shows that FABEO has the shortest setup time with 0.011s while it is 0.017s for Easy-ABE. For key generation, encryption and decryption our scheme performs better than FAME and FABEO. This efficiency comes from the fact that: (1) the user attributes set is converted into a single bit string therefore, the running time of our key generation algorithm is independent of the number of attributes; (2) with the AND policy, a singleton set of authorized attribute strings is used in our encryption algorithm and our decryption algorithm searches one element in a singleton set for decryption.

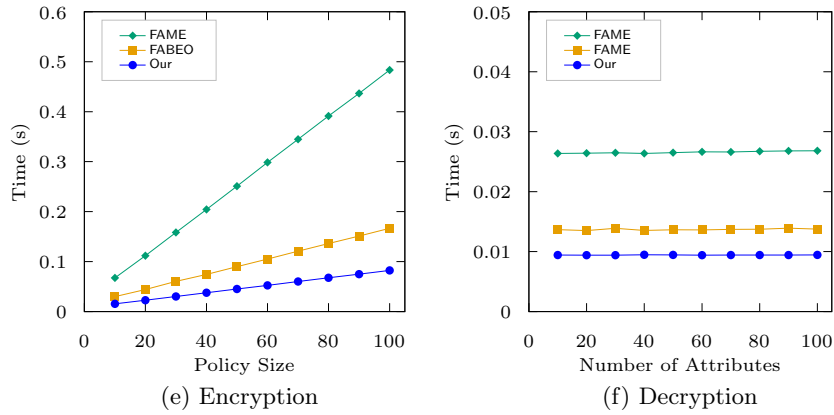


Fig. 3. Average running times for the encryption and decryption algorithms with OR policies. A random attribute in $[1..N]$ is used in decryption where N is the number of attributes. For the encryption we use a random 16 bytes plaintext.

Figure 3. The experiments performed with OR policies show that although our encryption algorithm uses a set of N authorized attribute strings and our decryption algorithm searches one element in a set of size N , Easy-ABE remains better than FAME and FABEO. From figure 3(f) we see that the cost of the search algorithm called in our decryption algorithm is negligible because for 10 attributes decryption takes 9.44ms and for 100 attributes it takes 9.45ms.

Since the running times of the algorithms are monotonic, table 5 gives a summary of the results of the experiments for 100 attributes. Additionally, we provide the storage cost in bytes of secret keys and ciphertexts for 100 attributes in Table 6.

Table 5. Average running times for the different algorithms for 100 attributes with the AND policy (top) and the OR policy (bottom).

Scheme	Running times (ms)			
	Setup	Keygen	Encrypt	Decrypt
FAME	25.64	682.96	530.84	29.12
FABEO	11.57	80.70	171.46	14.82
Our	16.89	6.77	8.71	9.28
FAME	–	–	483.44	26.81
FABEO	–	–	166.76	13.76
Our	–	–	82.47	9.45

Table 6. Comparison of secret key and ciphertext storage cost for 100 attributes with the AND policy (top) and the OR policy (bottom). For the encryption we use a random 1 KiB (1024 bytes) plaintext.

Scheme	Storage cost (bytes)	
	Key size	Ciphertext size
FAME	20800	23322
FABEO	7695	10300
Our	435	2261
FAME	1233	23319
FABEO	521	10279
Our	441	9983

These results are in line with the theoretical comparison carried out in section 5.

7 Conclusion

In this paper, we have proposed a ciphertext-policy attribute-based encryption scheme denoted Easy-ABE that is efficient in terms of computational and storage cost. Our scheme is non-monotonic but can be monotonic when it is accepted

for a user to query for a secret key associated to a subset of her/his set of attributes. Easy-ABE has the five essential properties required for a practical ABE scheme: it is based on the fast Type-3 pairings, is adaptively secure under CBDH-3 assumption, has unboundedness, large universe and fast decryption.

Easy-ABE was implemented in the Charm framework and proved to be better than FAME and FABEO.

With the use of DHIES, we believe that our scheme has indistinguishable encryptions under a chosen-ciphertext attack, but this remains to be proven. It would be interesting to prove it without resorting to the random oracle heuristic.

Acknowledgements. We would like to thank the anonymous reviewers for their detailed and insightful comments on an early draft of this paper.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of dhies. In: Naccache, D. (ed.) *Topics in Cryptology — CT-RSA 2001*. pp. 143–158. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
2. Agrawal, S.: Abe (2017), <https://github.com/sagrawal87/ABE>
3. Agrawal, S., Chase, M.: Fame: Fast attribute-based message encryption. CCS '17, Association for Computing Machinery, New York, NY, USA (2017), <https://doi.org/10.1145/3133956.3134014>
4. Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering* **3**(2), 111–128 (2013). <https://doi.org/10.1007/s13389-013-0057-3>
5. Ambrosin, M., Anzanpour, A., Conti, M., Dargahi, T., Moosavi, S.R., Rahmani, A.M., Liljeberg, P.: On the feasibility of attribute-based encryption on internet of things devices. *IEEE Micro* **36**(6), 25–35 (2016). <https://doi.org/10.1109/MM.2016.101>
6. Attrapadung, N., Herranz, J., Laguillaumie, F., Libert, B., de Panafieu, E., Ràfols, C.: Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science* **422**, 15–38 (2012). <https://doi.org/https://doi.org/10.1016/j.tcs.2011.12.004>
7. Bellare, M., Namprempe, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology* **21**(4), 469–491 (2008), <https://doi.org/10.1007/s00145-008-9026-x>
8. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP '07). pp. 321–334 (2007). <https://doi.org/10.1109/SP.2007.11>
9. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) *Advances in Cryptology – CRYPTO 2004*. pp. 443–459. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
10. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) *Advances in Cryptology — CRYPTO 2001*. pp. 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

11. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings - the role of ψ revisited. *Discrete Applied Mathematics* **159**(13), 1311–1322 (2011). <https://doi.org/https://doi.org/10.1016/j.dam.2011.04.021>
12. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded abe via bilinear entropy expansion, revisited. In: Nielsen, J.B., Rijmen, V. (eds.) *Advances in Cryptology – EUROCRYPT 2018*. pp. 503–534. Springer International Publishing, Cham (2018)
13. Chen, J., Wee, H.: Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In: Abdalla, M., De Prisco, R. (eds.) *Security and Cryptography for Networks*. pp. 277–297. Springer International Publishing, Cham (2014)
14. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. p. 456–465. *CCS '07*, Association for Computing Machinery, New York, NY, USA (2007), <https://doi.org/10.1145/1315245.1315302>
15. Doshi, N., Jinwala, D.C.: Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption. *Sec. and Commun. Netw.* **7**(11), 1988–2002 (nov 2014), <https://doi.org/10.1002/sec.913>
16. Edemacu, K., Jang, B., Kim, J.: Cescr: Cp-abe for efficient and secure sharing of data in collaborative ehealth with revocation and no dummy attribute. vol. 16. *PLoS ONE* (2021), <https://doi.org/10.1371/journal.pone.0250992>
17. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Bao, F., Li, H., Wang, G. (eds.) *Information Security Practice and Experience*. pp. 13–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
18. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) *Advances in Cryptology – EUROCRYPT 2010*. pp. 44–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
19. Galbraith, S.D.: *Mathematics of Public Key Cryptography*. Cambridge University Press (2012). <https://doi.org/10.1017/CB09781139012843>
20. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. p. 89–98. *CCS '06*, Association for Computing Machinery, New York, NY, USA (2006), <https://doi.org/10.1145/1180405.1180418>
21. Guo, F., Mu, Y., Susilo, W., Wong, D.S., Varadharajan, V.: Cp-abe with constant-size keys for lightweight devices. *IEEE Transactions on Information Forensics and Security* **9**(5), 763–771 (2014). <https://doi.org/10.1109/TIFS.2014.2309858>
22. Herranz, J., Laguillaumie, F., Ràfols, C.: Constant size ciphertexts in threshold attribute-based encryption. In: Nguyen, P.Q., Pointcheval, D. (eds.) *Public Key Cryptography – PKC 2010*. pp. 19–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
23. Ka, A.K.: *Easyabe* (2022), <https://github.com/khoureich/EasyABE>
24. Kothari, R., Choudhary, N., Jain, K.: Cp-abe scheme with decryption keys of constant size using ecc with expressive threshold access structure. In: Mathur, R., Gupta, C.P., Katewa, V., Jat, D.S., Yadav, N. (eds.) *Emerging Trends in Data Driven Computing and Communications*. pp. 15–36. Springer Singapore, Singapore (2021)
25. Krawczyk, H.: Cryptographic extraction and key derivation: The hkdf scheme. In: Rabin, T. (ed.) *Advances in Cryptology – CRYPTO 2010*. pp. 631–648. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

26. Kumar, N.S., Lakshmi, G.R., Balamurugan, B.: Enhanced attribute based encryption for cloud computing. *Procedia Computer Science* **46**, 689–696 (2015). <https://doi.org/https://doi.org/10.1016/j.procs.2015.02.127>, proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India
27. Lewko, A., Waters, B.: Unbounded hibe and attribute-based encryption. In: Paterson, K.G. (ed.) *Advances in Cryptology – EUROCRYPT 2011*. pp. 547–567. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
28. Odelu, V., Das, A.K.: Design of a new cp-abe with constant-size secret keys for lightweight devices using elliptic curve cryptography. *Security and Communication Networks* **9**(17), 4048–4059 (2016). <https://doi.org/https://doi.org/10.1002/sec.1587>
29. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) *Advances in Cryptology – ASIACRYPT 2012*. pp. 349–366. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
30. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. p. 195–203. *CCS '07*, Association for Computing Machinery, New York, NY, USA (2007), <https://doi.org/10.1145/1315245.1315270>
31. Papanis, J.P., Papapanagiotou, S.I., Mousas, A.S., Lioudakis, G.V., Kaklamani, D.I., Venieris, I.S.: On the use of attribute-based encryption for multimedia content protection over information-centric networks. *Transactions on Emerging Telecommunications Technologies* **25**(4), 422–435 (2014), <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.2722>
32. Riepel, D.: *Fabeo* (2023), <https://github.com/DoreenRiepel/FABEO>
33. Riepel, D., Wee, H.: *Fabeo*: Fast attribute-based encryption with optimal security. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. p. 2491–2504. *CCS '22*, Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3548606.3560699>, <https://doi.org/10.1145/3548606.3560699>
34. Rogaway, P.: Evaluation of some blockcipher modes of operation. *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan* (2011)
35. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. p. 463–474. *CCS '13*, Association for Computing Machinery, New York, NY, USA (2013), <https://doi.org/10.1145/2508859.2516672>
36. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) *Advances in Cryptology – EUROCRYPT 2005*. pp. 457–473. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
37. Tomida, J., Kawahara, Y., Nishimaki, R.: Fast, compact, and expressive attribute-based encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) *Public-Key Cryptography – PKC 2020*. pp. 3–33. Springer International Publishing, Cham (2020)
38. Zhang, Y., Zheng, D., Chen, X., Li, J., Li, H.: Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) *Provable Security*. pp. 259–273. Springer International Publishing, Cham (2014)

39. Zhao, Y., Xie, X., Zhang, X., Ding, Y.: A revocable storage cp-abe scheme with constant ciphertext length in cloud storage. *Mathematical Biosciences and Engineering* **16**(5), 4229–4249 (2019), <https://www.aimspress.com/article/doi/10.3934/mbe.2019211>
40. Zhou, Z., Huang, D.: On efficient ciphertext-policy attribute based encryption and broadcast encryption: Extended abstract. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. p. 753–755. CCS '10, Association for Computing Machinery, New York, NY, USA (2010), <https://doi.org/10.1145/1866307.1866420>