International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 14, Number 2, pages 186–205, July 2024

ELiPS-based Ciphertext-Policy Attribute-Based Encryption

Le Hoang Anh[*†], Yuta Kawada[*],
Samsul Huda[‡], Md. Arshad Ali[§], Yuta Kodera[*], and Yasuyuki Nogami[*]
[*]Graduate School of Environmental, Life, Natural Science and Technology, Okayama University, Japan
[‡]Green Innovation Center, Okayama University, Japan
{shuda, yuta_kodera, yasuyuki.nogami}@okayama-u.ac.jp
{lhanh, yuta_kawada}@s.okayama-u.ac.jp
[§]Faculty of CSE, Hajee Mohammad Danesh Science and Technology University, Bangladesh
arshad@hstu.ac.bd
[†]An Giang University, Vietnam National University Ho Chi Minh City, Vietnam
lhanh@agu.edu.vn

Received: February 15, 2024
Revised: May 5, 2024
Accepted: June 4, 2024
Communicated by Toru Nakanishi

Abstract

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) as an advanced cryptographic method keeps data safe in places like cloud storage and the Internet of Things using pairing-based cryptography. However, it relies on an outdated pairing-based cryptography (PBC) library, making it vulnerable to various attacks. Moreover, it does not meet today's demand for top-level security. Besides, the Efficient Library for Pairing Systems (ELiPS) offers efficient operations related to pairing-based cryptography, delivering high performance while upholding a substantial security standard. To deal with the shortcomings of CP-ABE, we adopt and implement the ELiPS as an efficient library for pairing systems into the CP-ABE framework, namely ELiPS-based CP-ABE. However, CP-ABE requires symmetric pairing, while ELiPS offers asymmetric pairing. To bridge this gap, our approach involves generating a generator g to transform asymmetric to symmetric pairing using Shirase's method, enabling compatibility between ELiPS and CP-ABE. Subsequently, we make several modifications to the CP-ABE framework and choose the appropriate ELiPS functions for integration. Finally, we validate our proposal through experiments involving data access authorization scenarios. Comparing with PBC-based CP-ABE, our ELiPS-based solution demonstrates reduced computational costs across most functions, except decryption. Additionally, our ELiPS-based CP-ABE performs comparably to the competitive MCL library, showcasing its efficiency and effectiveness in modern cryptographic applications.

Keywords: Attribute-Based Encryption (ABE), Ciphertext-Policy Attribute-Based Encryption (CP-ABE), Pairing-based cryptography (PBC), ELiPS, ELiPS-based CP-ABE

[0]This paper represents an improved and extended version of the work originally presented at 2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW).

1 Introduction

The Ciphertext-Policy Attribute-Based Encryption [1] is an advanced cryptographic protocol that safeguards privacy data in environments such as cloud storage [2] and the Internet of Things (IoT) [3]. Data is encrypted and protected based on an access policy. Only users who possess keys with attributes that satisfy the access policy can access and decrypt the encrypted data.

Cloud computing enables the storage and remote access of data via the internet. However, issues with access control and privacy arise when data is stored by a third party. On the other hand, IoT is a rapidly developing technology in the modern digital era. The large amounts of data generated by the expanding IoT have led to a greater focus on privacy and data access control in security. To meet these requirements, CP-ABE is utilized to provide privacy and fine-grained access control in both cloud storage [4, 5, 6, 7, 8] and IoT applications [9, 10, 11, 12, 13].

However, the CP-ABE employed on the PBC library [14], which has not been updated for a significant period and lacks sufficient security strength, may pose a potential weakness in modern cryptographic applications. The PBC library supports only an 80-bit security level, rendering it vulnerable to various attacks and limiting its practicality. Bos et al. [15] recommend that transitioning to a security level greater than 80-bit is necessary. According to Barker [16], an 80-bit of security is no longer regarded as being sufficiently secure.

On the other hand, the ELiPS[1] library provides efficient calculation costs while ensuring high security. It is a specialized cryptographic library that concentrates on efficient operations related to pairing-based cryptography [17]. ELiPS utilizes the BLS-12 curve and offers a 128-bit security level [18]. It provides several functions that support the implementation of algorithms and protocols that utilize pairing. Additionally, the ELiPS library has not only been used for the implementation of Pairing-based Homomorphic Encryption by Kanenari et al. [19] but also has been utilized in the implementation of zk-SNARKs.

To deal with these challenges, the authors propose an ELiPS-based CP-ABE scheme, integrating ELiPS into the CP-ABE framework [20]. However, the integration process is not straightforward due to differences between PBC and ELiPS libraries, including function parameters, data types, and the type of pairing. Notably, ELiPS supports asymmetric pairing, while the original CP-ABE relies on symmetric pairing. To bridge this gap and ensure compatibility, we generate a generator g with the specific purpose of converting asymmetric pairing to symmetric pairing. This conversion is crucial for maintaining compatibility with the original CP-ABE scheme. We accomplish this conversion by employing Shirase's technique [21]. After that, we make essential modifications to ensure the integration of ELiPS into the CP-ABE framework. These modifications span across the setup, key generation, encryption, and decryption algorithms.

To validate the proposal, the authors conducted several experiments, utilizing a data access authorization process at the university level with various attribute policy scenarios. The analysis and experimental results demonstrate the effectiveness of the proposal, revealing not only an increase in the security level but also a reduction in computational requirements for the setup, key generation, and encryption functions, except for the decryption cost. Moreover, we compare our ELiPS-base solution with MCL to see if it works as well and stays up to date with today's security needs. This comparison confirms that our approach meets today's security standards while operating efficiently.

The following are some of this paper's main contributions:

- Introduce generator g over $E(\mathbb{F}_{p^{12}})$ specifically designed to convert asymmetric pairing to symmetric pairing, which is a significant contribution in addressing the compatibility issue between ELiPS and the original CP-ABE scheme.

- Employ Shirase's technique [21] to accomplish the conversion of asymmetric pairing to symmetric pairing using the generated generator g.

- Make several modifications to the CP-ABE framework and carefully select appropriate ELiPS functions to ensure compatibility of ELiPS with CP-ABE.

[1] *ELiPS*. Information Security laboratory Okayama University. https://github.com/ISecOkayamaUniv/ELiPS

- Evaluate the proposal in terms of computation time.
- Compare our proposal's performance with other competitive pairing libraries.

2 Preliminaries

In this section, we provide background information on arithmetic operations over the elliptic curve, all of which play vital roles in the CP-ABE algorithm. The authors briefly introduce the Discrete Logarithm Problem, Elliptic Curve Discrete Logarithm Problem, and access tree. Following that, we show an overview of CP-ABE, PBC, RELIC, MCL, and ELiPS. Additionally, we present a comparison between four prominent libraries in terms that are mainly utilized in the CP-ABE scheme.

2.1 Arithmetic over the elliptic curve

An elliptic curve E of short Weierstrass form defined over $\mathbb{F}_{p^m}$ is presented as follows [22]:

$$E : y^2 = x^3 + ax + b, \tag{1}$$

where m is an extension degree, a and b are coefficients in $\mathbb{F}_{p^m}$ satisfying following condition:

$$4a^3 + 27b^2 \neq 0.$$

The pair (x, y) that satisfies Equation (1) is called a rational point of E. $\#E(\mathbb{F}_{p^m})$ is number of rational points of $E(\mathbb{F}_{p^m})$: $\#E(\mathbb{F}_{p^m}) = p^m + 1 - t_m$, where $| t_m | \leq 2\sqrt{p^m}$. Let r be the largest prime factor that divides $\#E(\mathbb{F}_{p^m})$. Then, k be the minimal integer such that satisfies $r \mid (p^k - 1)$, which is called embedding degree.

Let $P = (x_P, y_P), Q = (x_Q, y_Q)$, and $R = (x_R, y_R)$ be affine rational points on E, as can be seen in Equation (1). The arithmetic operations over the elliptic curve are defined as follows.

- Elliptic Curve Addition (ECA)

  If $P \neq Q$, point addition formula for computing $R = P + Q$ is given as:

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P},$$
$$\begin{cases} x_R = \lambda^2 - x_P - x_Q, \\ y_R = \lambda(x_P - x_R) - y_P. \end{cases}$$

- Elliptic Curve Doubling (ECD)

  If $P = Q$, point doubling formula for computing $R = P + Q = P + P = 2P$ is given as follows:

$$\lambda = \frac{3x_P^2 + a}{2y_P},$$
$$\begin{cases} x_R = \lambda^2 - 2x_P, \\ y_R = \lambda(x_P - x_R) - y_P. \end{cases}$$

- Elliptic curve Scalar Multiplication (SCM)

  Repeating to use $+$ for P leads to the definition of a point sP, which is P multiplied by s. Point scalar multiplication formula for calculating $R = sP$ as:

$$R = sP = \underbrace{P + P + \cdots + P}_{s\text{-1 times additions.}}.$$

2.1.1 Hash function $\mathcal{H}$ onto elliptic curve

Hash function $\mathcal{H}$ maps any attribute described as a binary string to a random group element.

$$\mathcal{H} : \{0,1\}^* \to \mathbb{G}.$$

Hash function $\mathcal{H}$ has the following properties [23]:

- Pre-image resistance: For a given output h, it is computationally infeasible to find a value m such that $\mathcal{H}(m) = h$.
- 2nd pre-image resistance: For a given input m, it is computationally infeasible to find a value m', where $m \neq m'$ such that $\mathcal{H}(m) = \mathcal{H}(m')$.
- Collision resistance: It is computationally infeasible to find two values m and m', where $m \neq m'$ such that $\mathcal{H}(m) = \mathcal{H}(m')$.

2.1.2 Pairings on elliptic curve

The subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$ of $E(\mathbb{F}_{p^{12}})$ are defined as follows [21]:

$$\begin{cases} \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_p - [1]), \\ \mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi_p - [p]), \end{cases}$$

where $E[r]$ is a subgroup of order r on an elliptic curve over $\mathbb{F}_{p^{12}}$; $\pi_p : A \mapsto A^p$ is called a Frobenius endomorphism, a low-cost mapping for calculating p-th powering; $\mathrm{Ker}(\varphi)$ is the set of points mapped to the point at infinity $\mathcal{O}$ by the specified map φ: $\mathrm{Ker}(\varphi) = \{P \in E(\mathbb{F}_{p^{12}}) : \varphi(P) = \mathcal{O}\}$.

A pairing e is a map from two elements in groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to an element in group $\mathbb{G}_T$, defined as:

$$e : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T,$$

which has the following properties:

- Bilinear map

  For all rational points $P \in \mathbb{G}_1$, and $Q, Q' \in \mathbb{G}_2$, and integers $a, b \in \mathbb{Z}_r$, we have:

$$\begin{aligned} e(Q + Q', P) &= e(Q, P) \cdot e(Q', P), \\ e(aQ, bP) &= e(bQ, aP) = e(Q, P)^{ab}. \end{aligned}$$

- Non-degeneracy

  For all $P \neq \mathcal{O}_{\mathbb{G}_1}$ and $Q \neq \mathcal{O}_{\mathbb{G}_2}$, then:

$$e(Q, P) \neq 1.$$

2.1.3 Types of pairings

The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve subgroups, and the group $\mathbb{G}_T$ is the multiplicative group of a finite field. There are three types of pairings [23]:

- Type I: When $\mathbb{G}_1 = \mathbb{G}_2$.
- Type II: When $\mathbb{G}_1 \neq \mathbb{G}_2$ but an efficiently computable isomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ is known, while none is known in the other direction.
- Type III: When $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction.

Pairing Type I is also referred to as symmetric pairing while pairing Types II and III are known as asymmetric pairings.

2.1.4 Sextic twist

The element of $\mathbb{G}_2$ is a rational point in $E(\mathbb{F}_{p^{12}})$. However, it is known to only possess an equal amount of information with a rational point existing on $E'(\mathbb{F}_{p^2})$. Let z be a quadratic non-residue and cubic non-residue over $\mathbb{F}_{p^2}$ and defines two elliptic curves as follows:

$$\begin{cases} E : y^2 = x^3 + b & \text{over } \mathbb{F}_{p^{12}}, \\ E' : y^2 = x^3 + bz & \text{over } \mathbb{F}_{p^2}. \end{cases}$$

The sextic twist $\phi : E' \to E$ is defined as follows [23]:

$$\phi : E' \to E, \qquad (x, y) \mapsto (z^{-\frac{1}{3}}x, z^{-\frac{1}{2}}y).$$

2.2 Discrete Logarithm Problem and Elliptic Curve Discrete Logarithm Problem

The security of pairing-based cryptography is based on the difficulty of solving the Discrete Logarithm Problem (DLP) and the Elliptic Curve Discrete Logarithm Problem (ECDLP) [23].

In a finite field group, computing $a = b^x$ (where $a, b \in \mathbb{F}_p$, and $x \in \mathbb{Z}$) is straightforward, but determining x from given a and b poses a challenging problem known as the DLP.

Within the context of an elliptic curve group, obtaining $R = sP$ (where $P, R \in E(\mathbb{F}_p)$, and $s \in \mathbb{Z}$) is easily accomplished. However, the reverse process of deducing s from P and R is a complex problem referred to as the ECDLP.

2.3 Access tree

In this section, we introduce an access tree, which plays a crucial role in access control for CP-ABE. Then, we present how to check if a user's attributes match an access tree and provide an example.

2.3.1 Define an access tree

An access tree is used to describe the access policy of an encrypted message. For example, Figure 1 gives information about the access tree, which expresses the access policy as follows: (*Position: Professor* **OR** *Position: Researcher* **OR** *Position: Student*) **AND** (*Faculty: Engineering* **OR** *Faculty: Technology*).

Each non-leaf node of the access tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a non-leaf node x and k_x is its threshold value, then $0 < k_x \leq num_x$. For instance, two particular cases are **AND** and **OR** gates:

- **AND** gate: $k_x = num_x$.
- **OR** gate: $k_x = 1$.

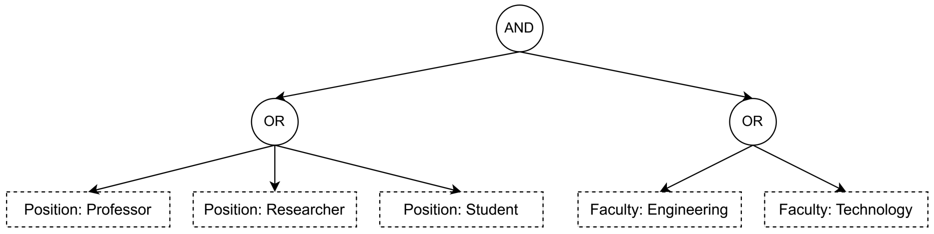

Figure 1: An example of a simple access tree $\mathcal{T}$.

Every leaf node x of the access tree is described by an attribute and a threshold value $k_x = 1$.

Some functions are defined to facilitate working with access trees:

- par(x) : denotes the parent of the node x in the tree.
- att(x) : is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree.
- ind(x) : denotes the order of the node x between its brothers. The nodes are randomly numbered from 1 to num.

2.3.2 Satisfying an access tree

Let $\mathcal{T}$ be an access tree with root r. $\mathcal{T}_x$ denotes the subtree of $\mathcal{T}$, which has root at the node x. Hence the tree $\mathcal{T}$ is the same as the $\mathcal{T}_x$. If a set of attributes A satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(A) = 1$, where A is a set of attributes, which is associated with the user's secret key. We compute $\mathcal{T}_x(A)$ recursively as follows:

- If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(A)$ for all children x' of node x. $\mathcal{T}_x(A)$ returns 1 if and only if at least k_x children return 1.
- If x is a leaf node, then $\mathcal{T}_x(A)$ returns 1 if and only if att$(x) \in A$.

For instance, if the receiver/decryptor possesses a secret key with the attribute set {*Position: Researcher, Faculty: Engineering*}, it satisfies the access tree as described in Figure 1. However, if the receiver/decryptor possesses a secret key with the attribute set {*Position: Researcher, Faculty: Agriculture*}, it does not satisfy the access tree as described in Figure 1.

2.4 Overview of Ciphertext-Policy Attribute-Based Encryption

CP-ABE is an encryption scheme that provides fine-grained access control over encrypted data. In CP-ABE, data is encrypted based on a set of attributes, and access to the encrypted data is granted based on predefined access policies associated with those attributes [20]. This approach allows for flexible and customizable access control, where data owners can define specific attributes required for decryption [24].

CP-ABE offers several advantages in scenarios where access control needs to be managed carefully. It enables data sharing among multiple users or organizations while ensuring that the data can only be accessed by those with the necessary credentials. The usage of CP-ABE is particularly relevant in cloud services, Internet of Things environments, and scenarios involving sensitive data storage and communication [25]. By leveraging attribute-based encryption, CP-ABE offers robust protection of data confidentiality and privacy [26]. It allows for secure data sharing, collaboration, and compliance with regulatory requirements.

The original CP-ABE implementation is based on the PBC library. In this work, we refer to it as PBC-based CP-ABE. The CP-ABE algorithm primarily relies on hash-to-curve and pairing procedures, comprising four main components.

2.4.1 Setup

The setup primitive is executed only once by the trusted party/server in the initial phase. This phase mainly uses scalar multiplication, pairing, and exponentiation operations for the computations. It outputs the master key MK and public key PK. Whereas the master key MK is kept secret, the public key PK is shared with all participants.

The algorithm begins by generating the $\mathbb{G}$ and $\mathbb{G}_T$ groups, where $\mathbb{G}$ has a generator g and both groups have an order r. Next, it randomly generates values α and $\beta \in \mathbb{Z}_r$. Then the master key MK and public key PK are calculated as follows [1]:

$$MK = (\beta, g^{\alpha}),$$
$$PK = (\mathbb{G}, g, h, f, v),$$

where: $h = g^{\beta}, f = g^{\beta^{-1}}, v = e(g, g)^{\alpha}, e$ is a bilinear map: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

2.4.2 Key generation

The key generation algorithm is also run once by the trusted party/server for each user. This phase primarily includes scalar multiplication and hash-to-curve operations. The algorithm takes the master key MK as well as the attribute set $A = \{att_1, att_2, ...\}$ as input. It proceeds to calculate the secret key SK, which is associated with the set of attributes A.

Firstly, the algorithm selects a random value $\gamma \in \mathbb{Z}_r$. Secondly, for each attribute $i \in A$, it selects a random value $\gamma_i \in \mathbb{Z}_r$. This part utilizes a hash function $\mathcal{H}$ to map each attribute into an element in $\mathbb{G} : \mathcal{H} = \{0,1\}^* \to \mathbb{G}$. Subsequently, the secret key SK is computed as [1]:

$$SK = (D, \{D_i, D'_i\}_{\forall i \in A}),$$

where: $D = g^{(\alpha+\gamma)\beta^{-1}}, D_i = g^{\gamma}\mathcal{H}(i)^{\gamma_i}, D'_i = g^{\gamma_i}$.

2.4.3 Encryption

This activity is executed by the sender/encryptor, who encrypts data on their devices. It primarily involves scalar multiplication and hash-to-curve operations. The encryption algorithm takes as input the public key PK, a message M, and an access policy $\mathcal{T}$ over the universe of attributes. It will encrypt message M and output a ciphertext CT such that only the receiver/decryptor who possesses a set of attributes associated with their secret key SK that satisfies the access tree $\mathcal{T}$ will be able to decrypt the message.

The encryption process is run as follows. A polynomial q_t is chosen for each node t in the access tree $\mathcal{T}$. The process chooses a random value $s \in \mathbb{Z}_r$, starting with the root R node, setting $q_R(0) = s$. Then, for every node $t \in \mathcal{T}$, it sets $q_t(0) = q_{\text{par}(t)}(\text{ind}(t))$. The leaf nodes in $\mathcal{T}$ are denoted as $\mathcal{L}$, and the function $\text{att}(t)$ provides the attribute value of each leaf node in the access tree. The message M is encrypted using the access policy $\mathcal{T}$, as follows [1]:

$$CT = (\mathcal{T}, \tilde{C}, C, \{C_l, C'_l\}_{\forall l \in \mathcal{L}}),$$

where: $\tilde{C} = Me(g,g)^{\alpha s}, C = h^s, C_l = g^{q_l(0)}, C'_l = \mathcal{H}(\text{att}(l))^{q_l(0)}$.

2.4.4 Decryption

The algorithm is run by the receiver/decryptor to decrypt the encrypted message on the server. The server will check whether the user's attributes satisfy the access policy. If the user's attributes match the access policy, the decryption process is successful, and the user gains access to the message; otherwise, they are denied access. This stage primarily employs pairing and multiplication operations for computations. It takes as input ciphertext CT, which contains an access policy $\mathcal{T}$, and a secret key SK constructed from a list A of attributes. If the set A of attributes satisfies the access tree $\mathcal{T}$ then the algorithm will be able to decrypt the ciphertext and return a message M.

The algorithm computes $\text{dec\_node}(CT, SK, t)$, which receives CT, SK, and node t as input. If t is a leaf node, the attribute of node t is obtained as $i = \text{att}(t)$. Then, $\text{dec\_node}(CT, SK, t)$ is computed as [1]:

$$\text{dec\_node}(CT, SK, t) = \begin{cases} \frac{e(D_i, C_t)}{e(D'_i, C'_t)} & \text{if } i \in A, \\ null & \text{if } i \notin A. \end{cases}$$

The $\text{dec\_node}(CT, SK, t)$ function operates on leafless node t as follows: For each child node c of t, the algorithm calls $\text{dec\_node}(CT, SK, c)$ and stores the result in F_c. A_t is a list of children c, where $F_c \neq null$. If no such set exists, the function returns *null*. Otherwise, the following calculation is performed [1]:

$$\text{Let: } k = \text{ind}(c), \qquad A'_t = \{\text{ind}(c), \forall c \in A_t\},$$

$$\Delta_{k, A'_t(0)} = \prod_{j \in A'_t, j \neq k} \frac{-j}{k-j},$$

$$F_t = \prod_{c \in A_t} F_c^{\Delta_{k,A'_t(0)}} = \prod_{c \in A_t} \left(e(g,g)^{\gamma q_c(0)}\right)^{\Delta_{k,A'_t(0)}} = \prod_{c \in A_t} \left(e(g,g)^{\gamma q_{\mathrm{par}(c)}(\mathrm{ind}(c))}\right)^{\Delta_{k,A'_t(0)}}$$
$$= \prod_{c \in A_t} e(g,g)^{\gamma q_t(k) \Delta_{k,A'_t(0)}} = e(g,g)^{\gamma q_t(0)}.$$

If the set of attributes A match the tree access policy $\mathcal{T}$, the algorithm then calls the dec_node(CT, SK, R) function as follows [1]:

$$\tilde{A} = \text{dec\_node}(CT, SK, R) = e(g,g)^{\gamma q_R(0)} = e(g,g)^{\gamma s}.$$

Then, the ciphertext is decrypted using the following formula [1]:

$$\frac{\tilde{C}}{\frac{e(C,D)}{\tilde{A}}} = M.$$

2.5 Overview of efficient libraries for pairing encryption

In this section, the authors present an overview of the PBC, RELIC, MCL, and ELiPS libraries as competitive pairing libraries. Then we demonstrate a comparison among four notable libraries in terms that are mainly utilized in the CP-ABE scheme.

2.5.1 Pairing-Based Cryptography (PBC) library

The GNU Multiple Precision (GMP) arithmetic library served as the foundation for the PBC library, an open source library carrying out the essential mathematical operations in pairing-based cryptosystems [27]. Speed and portability are crucial considerations as the PBC library is intended to serve as the foundation for pairing-based cryptosystem implementations. It offers functions like pairing computation and elliptic curve arithmetic [27].

In PBC, which utilizes symmetric pairing, let $\mathbb{G}$ be an additive group over an elliptic curve and $\mathbb{G}_T$ be a multiplicative cyclic group. Both groups $\mathbb{G}$ and $\mathbb{G}_T$ have order r [1]. The pairing operation is defined as:

$$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T.$$

There are eight different parameter types available in PBC. In each case, the curve group has a group order of 160-bit. Type A is known to be the fastest pairing and is suitable for cryptosystems where the group size is not a critical factor [14]. However, this type only provides an 80-bit security level and is vulnerable to multiple attacks [14, 15, 16]. Type A utilizes a supersingular curve, which is defined as follows:

$$E : y^2 = x^3 + x.$$

2.5.2 Efficient LIbrary for Cryptography (RELIC)

RELIC is an Efficient LIbrary for Cryptography, developed by Aranha et al. [28]. The first version was released in 2010. It is a contemporary cryptographic library, prioritizing efficiency and adaptability. RELIC focuses on portability, including architecture-dependent code, flexible configuration, and maximum efficiency [28].

RELIC utilizes both BN curves and BLS curves for configuration options. It supports a wide range of security levels such as 128-bit, 192-bit, and 256-bit [29]. The RELIC library supports nearly all functions necessary for the implementation of CP-ABE, such as elliptic curve addition, elliptic curve scalar multiplication, inversion, hash-to-curve, and pairing.

2.5.3 MCL

The MCL library, developed by Mitsunari et al. [29], is a high-performance library specializing in cryptographic operations and multi-core computation. It offers efficient implementations of mathematical operations crucial for modern cryptography, including elliptic curve cryptography and pairing-based cryptography.

With a focus on optimization and parallelism, MCL leverages multi-core processors to achieve fast execution time, making it ideal for applications requiring high computational efficiency. MCL is compatible with various operating systems and hardware architectures [30]. By providing developers with robust and optimized cryptographic primitives, the MCL library serves as a valuable tool for building secure and efficient cryptographic systems and protocols. MCL supports the BLS curve with a 381-bit characteristic and an embedding degree of 12, providing a 128-bit security level.

2.5.4 Efficient Library for Pairing Systems (ELiPS)

The ELiPS library is a specialized cryptographic library that focuses on efficient operations related to pairing-based cryptography. Such cryptography involves mathematical pairings between points on elliptic curves. The ELiPS library offers a range of functionalities, including point arithmetic operations, exponentiation, and pairing computations [17, 18]. ELiPS is specifically designed to support bilinear pairing using the BLS-12 curve, providing a 128-bit security level [17, 18].

The ELiPS library was evaluated and verified by Takahashi et al. [18]. It has gained attention for its applications in advanced cryptographic schemes such as identity-based encryption, attribute-based encryption, and functional encryption. Furthermore, it has been applied not only in the realization of Pairing-based Homomorphic Encryption by Kanenari et al. [19] but also being employed in the implementation of zk-SNARKs. This library is currently in development with regular updates, suggesting its potential as a promising resource.

2.5.5 A comparison among prominent pairing libraries in terms of primary domains used in CP-ABE

We conduct a comparative analysis of four prominent libraries in this research area: PBC [14], RELIC [28], MCL [30], and ELiPS [31]. We evaluate them across various metrics including hash-to-curve, pairing, exponentiation, scalar multiplication domains, security level, type of pairing, etc.

Table 1: Comparison among pairing libraries

Parameters		**PBC**	**RELIC**	**MCL**	**ELiPS**
Security level		80-bit	128-bit	128-bit	128-bit
Hash-to-curve		3.2 [ms]	0.6 [ms]	0.3 [ms]	0.1 [ms]
Pairing		0.9 [ms]	2.6 [ms]	1.1 [ms]	2.2 [ms]
Exponentiation		0.1 [ms]	1.3 [ms]	0.8 [ms]	0.6 [ms]
Scalar multiplication	$\mathbb{G}_1$	1.2 [ms]	0.3 [ms]	0.3 [ms]	0.2 [ms]
	$\mathbb{G}_2$	1.2 [ms]	0.7 [ms]	0.4 [ms]	0.5 [ms]
Functions in	$\mathbb{G}_T$	Multiplication Power	Multiplication Exponentiation	Multiplication Power	Multiplication Exponentiation
	$\mathbb{G}_1, \mathbb{G}_2$	Multiplication Power	Addition Multiplication	Addition Multiplication	ECA SCM
Type of pairing		I	III	III	III
Updated		Jun 15, 2013	Jun 20, 2022	Apr 20, 2023	Apr 12, 2023

Our findings, as summarized in Table 1, shows that some important tasks like hash-to-curve, pairing, exponentiation, and scalar multiplication are slower in PBC and RELIC compared to MCL and ELiPS. Since our goal is to improve the CP-ABE method, relying on PBC, we do not compare it to RELIC. Instead, we adopt ELiPS as it shows promise in our tests. We then utilize ELiPS to

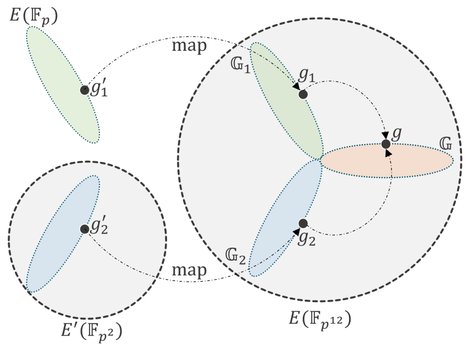

Figure 2: The process for generating g, g_1, and g_2.

enhance the CP-ABE method, calling it ELiPS-based CP-ABE. Our experiments demonstrate that ELiPS-based CP-ABE performs similarly to the MCL library, which is known for its effectiveness. This indicates that ELiPS could be a valuable option for enhancing this type of security method.

3 Proposed scheme

In this section, we present the main procedures required to implement CP-ABE using ELiPS, which we refer to as ELiPS-based CP-ABE. PBC and ELiPS use several different operations, as shown in Table 1. Therefore, we have designed three procedures to make ELiPS appropriate for CP-ABE. These procedures include generating g, g_1, and g_2, as well as transforming asymmetric to symmetric pairing and modifying CP-ABE framework functions.

3.1 Generator g generation

Generating a generator g serves the purpose of transforming asymmetric pairing, which is the basis of ELiPS, into symmetric pairing. This transformation is crucial for compatibility with the original CP-ABE, which relies on symmetric pairing.

Figure 2 illustrates the process of generating g. Firstly, the algorithm generates two generators g_1' and g_2' over $E(\mathbb{F}_p)$ and $E'(\mathbb{F}_{p^2})$, respectively. Then, g_1' and g_2' are mapped to g_1 in subgroup $\mathbb{G}_1$ and g_2 in subgroup $\mathbb{G}_2$, respectively. Since $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of $E[r]$, we can add the elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, and the result is an element of $E[r]$. Finally, the generator g of group $\mathbb{G}$ be generated as follows:

$$g = g_1 + g_2. \tag{2}$$

This method is significantly faster compared to directly creating a generator over $E(\mathbb{F}_{p^{12}})$. Since $\mathbb{G}$ is a subgroup of order r of $E[r](\subset E(\mathbb{F}_{p^{12}}))$, addition and scalar multiplication can be defined over $\mathbb{G}$ in the same way as those on $E(\mathbb{F}_{p^{12}})$ [21].

3.2 Asymmetric to symmetric transformation

The authors successfully implemented Shirase's method [21] for converting asymmetric pairing to symmetric pairing. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generator rational points. Then g is a generator point of $\mathbb{G}$, and this can be calculated as shown in Equation (2). For two rational points $P, Q \in \mathbb{G}$ and we can use symmetric pairing $e_{\text{sym}}(Q, P)$ by defining a symmetric pairing as follows [21]:

$$e_{\text{sym}} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T.$$

Since ELiPS uses asymmetric pairing, the authors need to transform asymmetric pairing into symmetric pairing. This is done by extracting P' in group $\mathbb{G}_1$ and Q' in group $\mathbb{G}_2$ from P and Q,

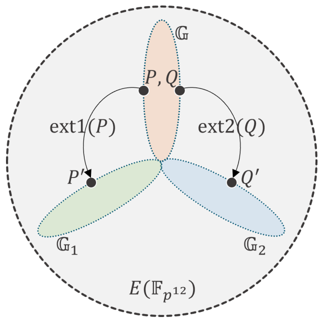

Figure 3: Extraction of P' and Q' in transforming asymmetric pairing to symmetric pairing.

respectively. Figure 3 shows the concept of the extraction procedure. The transformation between asymmetric and symmetric pairing can be defined as [21]:

$$e_{\text{sym}}(Q, P) = e_{\text{asy}}(\text{ext2}(Q), \text{ext1}(P)).$$

Next, the authors provide a method for extracting P' in group $\mathbb{G}_1$ and Q' in group $\mathbb{G}_2$ from P and Q in group $\mathbb{G}$, respectively.

Let $l = (p-1)^{-1} \pmod r$, where r is an order of subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$. Then, the values of ext1 and ext2 can be calculated as follows [21]:

$$\begin{cases} \text{ext1} = ([p] - \pi_p)[l], \\ \text{ext2} = (\pi_p - [1])[l]. \end{cases}$$

The symmetric pairing procedure is processed as follows:

- Calculate the rational points P and Q, where $P, Q \in \mathbb{G}$.
- Then, it calls the ext1 and ext2 functions to calculate P' and Q', as demonstrated in Figure 3.
$$P' = \text{ext1}(P),$$
$$Q' = \text{ext2}(Q),$$
where $P' \in \mathbb{G}_1$ and $Q' \in \mathbb{G}_2$.
- Afterward, the algorithm calls $e_{\text{asy}}(Q', P')$ to calculate asymmetric pairing. The asymmetric pairing function uses Miller loop and final exponentiation to calculate and return the pairing value.

3.3 CP-ABE algorithm modifications

We briefly present some modifications to enable ELiPS to work within the CP-ABE framework. Then, the authors conduct a security analysis on ELiPS-based CP-ABE, showing its alignment with today's security needs.

3.3.1 Setup

The setup is executed once on the server at the beginning of the system. The algorithm generates g'_1 and g'_2 over $E(\mathbb{F}_p)$ and $E'(\mathbb{F}_{p^2})$, respectively. Then, it maps g'_1 and g'_2 to g_1 and g_2 over $E(\mathbb{F}_{p^{12}})$, as illustrated in Figure 2. The generator point g is calculated using the formula $g = g_1 + g_2$. Next, random $\alpha, \beta \in \mathbb{Z}_r$ are generated. The master key MK and public key PK are computed as follows [20]:

$$\begin{aligned} MK &= (\beta, \alpha g). \\ PK &= (g, h, f, v), \end{aligned} \tag{3}$$

where: $h = \beta g, f = \beta^{-1} g, v = e(g, g)^{\alpha}$.

3.3.2 Key generation

This function is run once on the server for each user as well. The key generation takes the master key MK and attribute set A as input. It calculates the secret key SK as [20]:

$$SK = (D, \{D_i, D'_i\}_{\forall i \in A}), \tag{4}$$

where: $D = (\alpha + \gamma)\beta^{-1}g, D_i = \gamma g + \gamma_i \mathcal{H}(i), D'_i = \gamma_i g, \gamma$ and γ_i are random numbers over $\mathbb{Z}_r$.

3.3.3 Encryption

The encryption function is executed every time to encrypt data on the user's own devices. It takes the public key PK, message M, and access policy tree $\mathcal{T}$ as input. The ciphertext is computed as follows [20]:

$$CT = (\mathcal{T}, \tilde{C}, C, \{C_l, C'_l\}_{\forall l \in \mathcal{L}}), \tag{5}$$

where: $\tilde{C} = Me(g,g)^{\alpha s}, C = sh, C_l = q_l(0)g, C'_l = q_l(0)\mathcal{H}(\text{att}(l)), s$ is a random number over $\mathbb{Z}_r, \mathcal{L}$ is the leaf node set in $\mathcal{T}$.

3.3.4 Decryption

The decryption phase is run on the server. Firstly, it verifies the user's attributes and access policy. If the user's attributes meet the access policy, the decryption process is successful, and then the user can access the plain message. Otherwise, access is denied. The inputs for the procedure are ciphertext CT and secret key SK. It calls the dec_node(CT, SK, R) function to calculate $\tilde{A}$ as [20]:

$$\tilde{A} = \text{dec\_node}(CT, SK, R) = e(g,g)^{\gamma s}.$$

In this function, the algorithm calls the recursive dec_node(CT, SK, t) function to calculate the value $\tilde{A}$ and verify whether the secret key SK matches the access policy, where t is a leaf node, as follows [20]:

$$\text{dec\_node}(CT, SK, t) = \begin{cases} \frac{e_{\text{sym}}(D_i, C_t)}{e_{\text{sym}}(D'_i, C'_t)} & \text{if } i \in A, \\ null & \text{if } i \notin A. \end{cases} \tag{6}$$

The original message is decrypted using the following formula [20]:

$$\frac{\tilde{C}\tilde{A}}{e_{\text{sym}}(C, D)} = \frac{\tilde{C}\tilde{A}}{e_{\text{asy}}(\text{ext2}(C), \text{ext1}(D))} = M. \tag{7}$$

3.3.5 Security analysis

Numerous cryptographic protocols rely on computational assumptions to demonstrate their security. Mrabet et al. [23] noted that pairings of Type III are compatible with various computational assumptions, such as the Decision Diffie-Hellman in $\mathbb{G}_1$ or $\mathbb{G}_2$, also referred to as the External Diffie-Hellman assumption, which is not upheld in Type I pairings [20]. While the ELiPS-based CP-ABE relies on ELiPS, employing asymmetric pairing (Type III), the PBC-based CP-ABE utilizes symmetric pairing (Type I). This suggests that using asymmetric pairing is better than symmetric pairing from a security perspective.

Additionally, according to Equation (5) and Equation (7), to decrypt encrypted data, one needs to calculate the value of $e(g,g)^{\alpha s}$ or $e(C,D)/e(g,g)^{\gamma s}$, as follows:

- Recovering the value $e(g,g)^{\alpha s}$ requires attackers to determine α and s. However, based on the Discrete Logarithm Problem and the Elliptic Curve Discrete Logarithm Problem, computing α from $d = \alpha g$ or $v = e(g,g)^{\alpha}$ and s from $C = sh$ is infeasible.

- Calculating the value $e(C, D)/e(g, g)^{\gamma s}$ allows adversaries to compute $e(C, D)$ using C from the ciphertext and D from the user's secret key. However, the value of $e(g, g)^{\gamma s}$ remains blinded. Recovering γ from $D = (\alpha + \gamma)\beta^{-1}g$ and s from $C = sh$ are challenging problems, according to DLP and ECDLP.

  where $g, h \in E$, and $\alpha, \beta, \gamma, s \in \mathbb{Z}_r$.

Computing discrete logarithms is evidently difficult, which is related to the bit length of r. In PBC-based CP-ABE, r is 160 bits, while in ELiPS-based CP-ABE, it is 308 bits. This demonstrates that the proposed scheme increases the security level.

4 Experimental evaluation and discussion

In this section, we experiment and evaluate the performance of the proposal. Firstly, we evaluate the efficacy of setup, key generation, encryption, and decryption in PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE with a two-attribute scenario. Secondly, the authors validate the key generation, encryption, and decryption parts with an increasing number of attributes. Additionally, we further explore the total execution time, including setup time, key generation time, encryption time, and decryption time, across various scenarios.

4.1 Experimental evaluation setup

Table 2: Experimental environments

OS	Ubuntu 22.04.1 LTS (WSL2)
CPU	Intel(R) Core(TM) i7-6600U @ 2.60GHz
Memory	4 GB
Language	C
GMP version	6.2.1
GCC version	11.3.0
GCC optimization level	-O2

Table 2 shows the devices and software used during the evaluation. In our experiments, we employed a data access authorization for administration procedures at the university level and attribute policy scenarios, as depicted in Figure 4, which involve three entities:

- University administrator: Authority.
- President: Sender.
- Professors: Receiver.

Assuming the university president wishes to share private data exclusively with professors in the Faculty of Engineering, the president only encrypts the data once and shares the encrypted data with all intended recipients. The president also needs to define an access policy $\mathcal{T}$ structure to determine who can decrypt the encrypted data, as shown in Figure 4. On the recipients' side, if their attributes satisfy the access policy, they can successfully decrypt the data; otherwise, they are unable to decrypt it.

4.2 Performance evaluation with two-attribute scenario

We employed two attributes to implement a data access authorization for the administration scenario. We performed 10,000 executions to measure the computation time of setup, key generation,

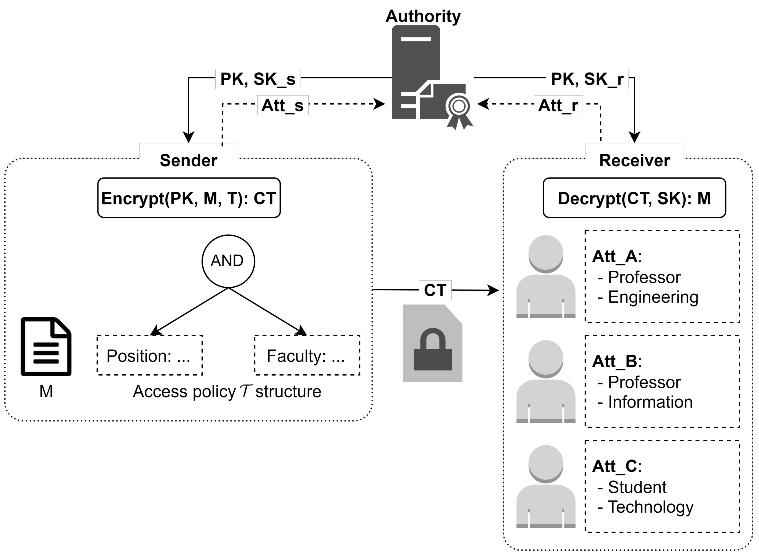

Figure 4: An example of data access authorization for administrative procedures at the university level [20].

encryption, and decryption functions for PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE, then we took the average values.

Table 3 summarizes the comparison results. Overall, it shows that most of the functions in ELiPS-based CP-ABE perform faster than their counterparts in PBC-based CP-ABE, except for the decryption function. The performance of both ELiPS-based CP-ABE and MCL-based CP-ABE are closely competitive, with no significant difference between them. It shows that our ELiPS-based solution is working well and operating efficiently.

Table 3: A comparison among PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE in a two-attribute scenario

Functions	**PBC-based CP-ABE**	**MCL-based CP-ABE**	**ELiPS-based CP-ABE**
Setup	5.6 [ms]	4.0 [ms]	4.1 [ms]
Keygen	15.5 [ms]	3.9 [ms]	3.8 [ms]
Encryption	15.0 [ms]	3.9 [ms]	3.7 [ms]
Decryption	7.3 [ms]	9.6 [ms]	11.0 [ms]

Table 3 shows that while the setup speed in ELiPS-based CP-ABE is faster than that in PBC-based CP-ABE by 26.78%, and MCL-based CP-ABE is faster than that in PBC-based CP-ABE by 28.57%. In addition, the data illustrates that the key generation performance in MCL-based CP-ABE is better than that in PBC-based CP-ABE by 74.84%, while the key generation in ELiPS-based CP-ABE is better than other schemes by 2.56% compared to MCL-based CP-ABE and by 75.48% compared to PBC-based CP-ABE.

Regarding the encryption part, Table 3 shows that encryption time in ELiPS-based CP-ABE is the best among three versions, namely PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE. Whereas encryption time in MCL-based CP-ABE decreases by 74.00%, encryption time in ELiPS-based CP-ABE reduces by 75.33% compared to that in PBC-based CP-ABE. On the other hand, the decryption time for MCL-based CP-ABE and ELiPS-based CP-ABE increases by 31.51% and 50.68%, respectively, compared to the decryption time for PBC-based CP-ABE. Therefore, further evaluation with increasing the number of attributes is necessary.

4.3 Evaluating the key generation, encryption, and decryption processes with an increasing number of attributes

Since the setup part is not affected by the number of attributes, we do not need to evaluate it further. Instead, we focus on experiments and evaluations of key generation, encryption, and decryption with varying numbers of attributes. Moreover, the authors explore the total execution time in three versions.

Firstly, the authors conducted experiments 10,000 times to measure the key generation time, encryption time, and decryption time in PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE. We then calculated the average results.

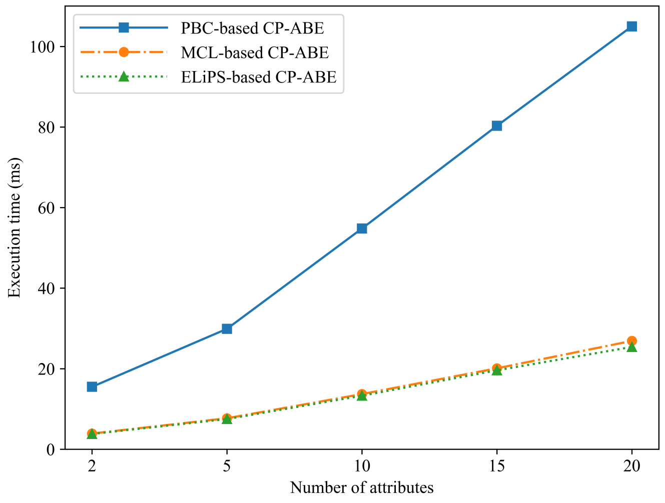

Figure 5: Key generation time for several scenarios of PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE.

Figure 5 shows the key generation performance in MCL-based CP-ABE is better than that in PBC-based CP-ABE by 74.68%, while the key generation in ELiPS-based CP-ABE is better than other schemes by 3.73% compared to MCL-based CP-ABE and by 75.62% compared to PBC-based CP-ABE. The increase in performance of key generation in MCL-based CP-ABE and ELiPS-based CP-ABE can be attributed to the fact that the key generation algorithm primarily utilizes the hash-to-curve and SCM operations for each attribute, as illustrated in Table 4. These operations exhibit superior performance in MCL and ELiPS compared to PBC, as indicated in Table 1.

Table 4: Computations cost in CP-ABE algorithm

Functions	**Computational cost**
Key generation	$1\mathcal{J} + n\mathcal{H} + (n+1)\mathcal{A} + 2(n+1)\mathcal{S}$
Encryption	$1\mathcal{M} + 1\mathcal{E} + n\mathcal{H} + (2n+1)\mathcal{S}$
Decryption	$2\mathcal{M} + 1\mathcal{J} + (2n+1)\mathcal{P}$

where: $\mathcal{M}$ is the multiplication cost over $\mathbb{F}_{p^{12}}$, $\mathcal{E}$ is the exponentiation cost over $\mathbb{F}_{p^{12}}$, $\mathcal{J}$ is the inversion cost over $\mathbb{F}_{p^{12}}$, $\mathcal{H}$ is the hash-to-curve cost, $\mathcal{P}$ is the pairing cost, $\mathcal{A}$ is the elliptic curve addition cost, $\mathcal{S}$ is the elliptic curve scalar multiplication cost, n is the number of attributes.

Figure 6 shows a similar trend to the key generation part. Encryption time in ELiPS-based CP-ABE is the best among the three versions. Encryption time in ELiPS-based CP-ABE decreases by 75.04% compared to that in PBC-based CP-ABE and reduces by 4.88% compared to that in MCL-based CP-ABE. Similar to the key generation algorithm, Table 4 shows that the encryption algorithm primarily utilizes hash-to-cure and SCM operations, which are employed for each attribute

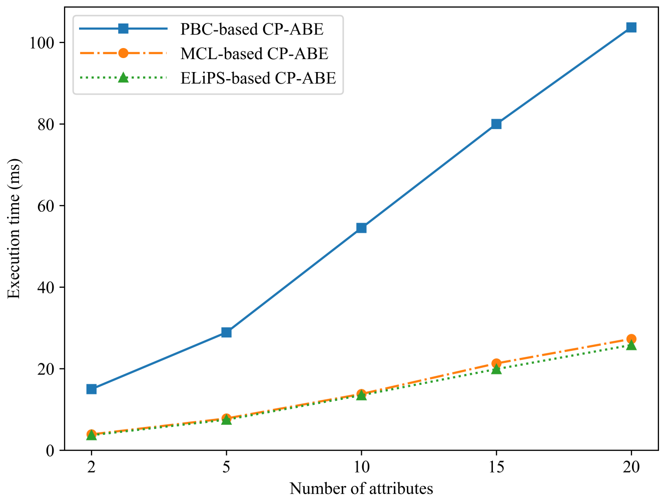

Figure 6: Encryption time for several scenarios of PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE.

in the access policy. The results in Table 1 indicate that the computational cost of these operations in MCL and ELiPS significantly reduces compared to that in PBC. Hence, the encryption time in the proposal decreases around 3.9-fold compared to that in PBC-based CP-ABE.

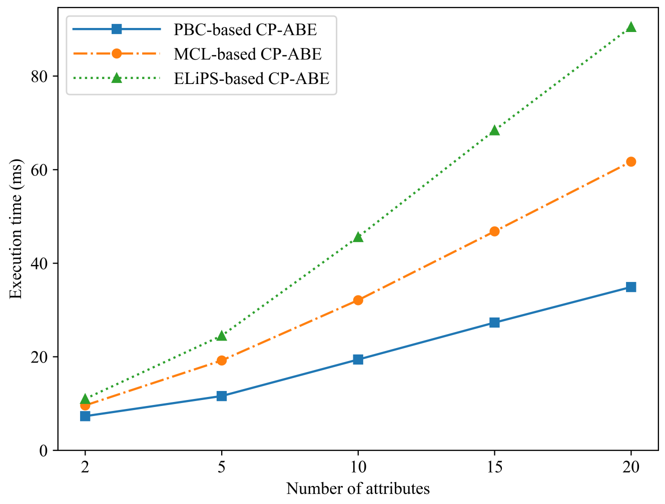

Figure 7: Decryption time for several scenarios of PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE.

Figure 7 demonstrates that the decryption time of both MCL-based CP-ABE and ELiPS-based CP-ABE is higher than that of the PBC-based CP-ABE across scenarios. The decryption time increases linearly as the number of attributes increases. As indicated in Table 4, the number of pairing operations depends on the number of attributes. Additionally, the pairing cost in both MCL and ELiPS is heavier than that in PBC, as demonstrated in Table 1.

The experimental results demonstrate that while setup time, key generation time, and encryption time in the proposal are lower than those in PBC-based CP-ABE, decryption time is higher. Thus, we would like to further explore the total execution time. The authors conducted the experiments 10,000 times to measure the total execution time, including setup time, key generation time, encryption time, and decryption time for PBC-based CP-ABE, MCL-based CP-ABE, and ELiPS-based CP-ABE. We then took the average results.

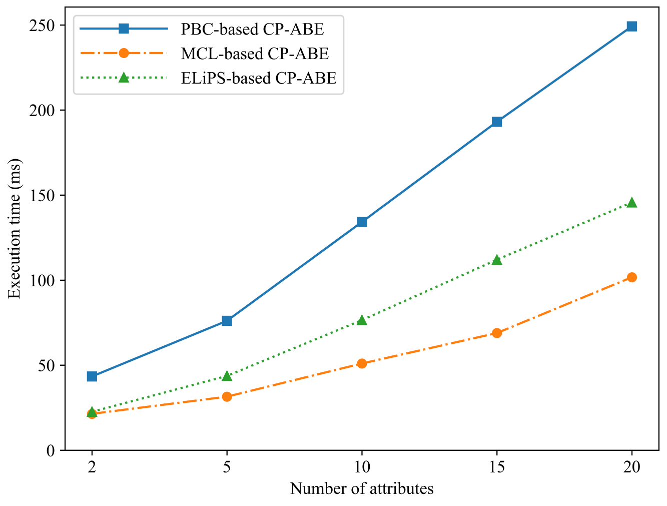

Figure 8: The execution time of four functions among three versions in several scenarios.

Figure 8 illustrates that the proposed scheme improved performance compared to the PBC-based CP-ABE. The total execution time in MCL-based CP-ABE and ELiPS-based CP-ABE is reduced by an average of 60.56% and 42.46%, respectively, compared to that in PBC-based CP-ABE.

In IoT scenarios, efficient encryption is crucial for resource-constrained IoT devices acting as senders. While the setup and key generation are one-time operations performed on the server, encryption and decryption must be executed repeatedly for securing and accessing data. According to the results in Table 3 and Figure 6, the ELiPS-based CP-ABE scheme outperforms other compared schemes like PBC-based CP-ABE and MCL-based CP-ABE in terms of encryption performance. Since IoT devices have limited resources, the superior encryption efficiency of ELiPS-based CP-ABE makes it suitable for practical implementation in IoT scenarios where the encryption is handled by the devices, and the server handles the decryption.

On the other hand, the PBC library utilizes symmetric pairing, while the ELiPS library uses asymmetric pairing (Type III). The symmetric pairing is not robust enough from a security point of view [23]. Pairings categorized as Type III align with various computational assumptions, including the Decision Diffie-Hellman assumption in $\mathbb{G}_1$ or $\mathbb{G}_2$, also referred to as the External Diffie-Hellman assumption, which does not hold in Type I pairings [23]. Therefore, the ELiPS-based solution is more compatible with various computational assumptions than PBC-based CP-ABE.

Moreover, the security levels for ELiPS and PBC are different. Comparing them will be more appropriate when PBC-based CP-ABE and ELiPS-based CP-ABE use the same security level.

5 Conclusion

We introduced an ELiPS-based CP-ABE scheme by integrating ELiPS as an efficient library for pairing systems into the CP-ABE encryption method. Here, generating a generator g served the purpose of transforming asymmetric pairing, which was the basis of ELiPS, into symmetric pairing. This transformation was crucial for compatibility with the original CP-ABE, which relied on symmetric pairing. Then, we transformed asymmetric pairing to symmetric pairing using Shirase's technique and made several modifications to the CP-ABE framework for the integration. The experimental results confirmed that the proposal not only improved the performance but also boosted the security level to 128 bits for CP-ABE. Additionally, comparing ELiPS-based CP-ABE with MCL showed that our ELiPS-based solution functioned efficiently while matching today's security needs. The superior encryption efficiency of ELiPS-based CP-ABE made it suitable for practical implementation in IoT scenarios where the encryption was handled by the devices, and the server handled the decryption. As future work, we will consider improving the decryption efficiency of the ELiPS-based CP-ABE.

References

[1] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *2007 IEEE Symposium on Security and Privacy (SP '07),* Berkeley, CA, USA, pp. 321–334, 2007, doi: 10.1109/SP.2007.11.

[2] Y. Lu, Y. Wang, X. Dai, J. Li, J. Li, and M. Chen, "Survey of Attribute-Based Encryption in Cloud Environment," *Communications in Computer and Information Science.* vol. 1127, pp. 375–384, 2020, doi: 10.1007/978-981-15-6113-9_43.

[3] M. Rasori, M. L. Manna, P. Perazzo, and G. Dini, "A Survey on Attribute-Based Encryption Schemes Suitable for the Internet of Things," *IEEE Internet of Things Journal,* vol. 9, no. 11, pp. 8269–8290, 2022, doi: 10.1109/JIOT.2022.3154039.

[4] S. Huda, A. Sudarsono, and T. Harsono, "Secure data exchange using authenticated ciphertext-policy attributed-based encryption," *2015 International Electronics Symposium (IES),* pp. 134–139, 2015, doi: 10.1109/ELECSYM.2015.7380829.

[5] S. Huda, A. Sudarsono, and T. Harsono, "Secure Communication and Information Exchange using Authenticated Ciphertext Policy Attribute-Based Encryption in Mobile Ad-hoc Network," *EMITTER International Journal of Engineering Technology,* vol. 4, no. 1, pp. 115–140, 2016, doi: 10.24003/emitter.v4i1.116.

[6] T. P. Ezhilarasi, N. S. Kumar, T. P. Latchoumi, and N. Balayesu, "A Secure Data Sharing Using IDSS CP-ABE in Cloud Storage," *Advances in Industrial Automation and Smart Manufacturing,* Singapore: Springer, Singapore, pp. 1073–1085, 2021, doi: 10.1007/978-981-15-4739-3_92.

[7] Y. W. Hwang and I. Y. Lee, "A Study on Lightweight Anonymous CP-ABE Access Control for Secure Data Protection in Cloud Environment," *2019 International Conference on Information Technology and Computer Communications (ITCC'19),* New York, NY, USA, pp. 107–111, 2019, doi: 10.1145/3355402.3355405.

[8] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based Encryption for Cloud Computing Access Control: A Survey," *ACM Comput. Surv,* vol. 53, no. 4, pp. 1–41, 2020, doi: 10.1145/3398036.

[9] S. Huda, N. Fahmi, A. Sudarsono, and M.U.H. Al Rasyid, "Secure data sensor sharing on ubiquitous environmental health monitoring application," *Jurnal Teknologi (Sciences & Engineering),* vol. 78, no. 6-3, pp. 53–58, 2016. doi: 10.11113/jt.v78.8928.

[10] R. Cheng, K. Wu, Y. Su, W. Li, W. Cui, and J. Tong, "An Efficient ECC-Based CP-ABE Scheme for Power IoT," *Processes 2021,* vol. 9, no. 1176, pp. 1–16, 2021, doi: 10.3390/pr9071176.

[11] B. Girgenti, P. Perazzo, C. Vallati, F. Righetti, G. Dini, and G. Anastasi, "On the Feasibility of Attribute-Based Encryption on Constrained IoT Devices for Smart Systems," *2019 IEEE International Conference on Smart Computing (SMARTCOMP),* Washington, DC, USA, pp. 225–232, 2019, doi: 10.1109/SMARTCOMP.2019.00057.

[12] P. Perazzo, F. Righetti, M. L. Manna, and C. Vallati, "Performance evaluation of Attribute-Based Encryption on constrained IoT devices," *Computer Communications,* vol. 170, pp. 151–163, 2021, doi: 10.1016/j.comcom.2021.02.012.

[13] D. Ziegler, J. Sabongui, and G. Palfinger, "Fine-Grained Access Control in Industrial Internet of Things," *IFIP Advances in Information and Communication Technology,* Springer, vol. 562, pp. 91–104, 2019, doi: 10.1007/978-3-030-22312-0_7.

[14] B. Lynn. Stanford University. *PBC Library - Pairing-Based Cryptography.* (2006). Accessed: Jul. 15, 2023. [Online]. Available: https://crypto.stanford.edu/pbc

[15] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery, "On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography," *Cryptology ePrint Archive,* pp. 1–19, 2009. [Online]. Available: https://eprint.iacr.org/2009/389

[16] E. Barker, "Recommendation for Key Management," *National Institute of Standards and Technology,* 2020, doi: 10.6028/NIST.SP.800-57pt1r5.

[17] D. Hattori, Y. Takahashi, T. Tatara, Y. Nanjo, T. Kusaka, and Y. Nogami, "An Optimal Curve Parameters for BLS12 Elliptic Curve Pairing and Its Efficiency Evaluation," in *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW),* Penghu, Taiwan, pp. 1–2, 2021, doi: 10.1109/ICCE-TW52618.2021.9602941.

[18] Y. Takahashi, Y. Nanjo, T. Kusaka, Y. Nogami, T. Kanenari, and T. Tatara, "An Implementation and Evaluation of Pairing Library ELiPS for BLS Curve with Several Techniques," *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC),* JeJu, Korea (South), 2019, doi: 10.1109/ITC-CSCC.2019.8793376.

[19] T. Kanenari, Y. Takahashi, Y. Hashimoto, Y. Kodera, T. Kusaka, Y. Nogami, and T. Nakanishi, "A Comparison of Relic-toolkit and ELiPS Libraries for a Pairing-based Homomorphic Encryption," *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC),* JeJu, Korea (South), 2019, doi: 10.1109/ITC-CSCC.2019.8793446.

[20] L. H. Anh, Y. Kawada, S. Huda, M. A. Ali, Y. Kodera, and Y. Nogami, "An implementation of ELiPS-based Ciphertext-Policy Attribute-Based Encryption," *2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW),* Matsue, Japan, pp. 220–226, 2023, doi: 10.1109/CANDARW60564.2023.00044.

[21] M. Shirase, "Symmetric Pairing on Ordinary Elliptic Curves," in *Information Processing Society of Japan Symposium Proceedings,* Japan, pp. 357–362, 2010.

[22] Y. Nanjo, M. Shirase, Y. Kodera, T. Kusaka, and Y. Nogami, "Efficient Final Exponentiation for Cyclotomic Families of Pairing-Friendly Elliptic Curves with Any Prime Embedding Degrees," *International Journal of Networking and Computing,* vol. 12, no. 2, pp. 317–338, 2022. [Online]. Available: http://www.ijnc.org/index.php/ijnc/article/view/285

[23] N. E. Mrabet and M. Joye, "Guide to Pairing-Based Cryptography," *Chapman and Hall/CRC,* pp. 1–420, 2016, doi: 10.1201/9781315370170.

[24] K. P. Praveen, K. P. Syam, and P. J. A. Alphonse, "Attribute based encryption in cloud computing: A survey, gap analysis, and future directions," *Journal of Network and Computer Applications,* vol. 108, pp. 37–52, 2018, doi: 10.1016/j.jnca.2018.02.009.

[25] B. Chandrasekaran, R. Balakrishnan, and Y. Nogami, "TF-CPABE: An efficient and secure data communication with policy updating in wireless body area networks," *ETRI Journal,* vol. 41, no. 4, pp. 465–472, 2019, doi: 10.4218/etrij.2018-0320.

[26] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute Based Encryption with Privacy Protection and Accountability for CloudIoT," *IEEE Transactions on Cloud Computing,* vol. 10, no. 2, pp. 762–773, 2022, doi: 10.1109/TCC.2020.2975184.

[27] J. Bethencourt, A. Sahai, and B. Waters. The University of Texas. *Advanced Crypto Software Collection.* (2006). Accessed: Jul. 15, 2023. [Online]. Available: https://acsc.cs.utexas.edu/cpabe

[28] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. Github. *RELIC is an Efficient LIbrary for Cryptography.* (2013). Accessed: Jan. 19, 2024. [Online]. Available: https://github.com/relic-toolkit/relic

[29] Y. Sakemi, T. Kobayashi, T. Saito, and R. Wahby, “Pairing-Friendly Curves,” The Internet Engineering Task Force (IETF), 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-irtf-cfrg-pairing-friendly-curves-10.html

[30] S. Mitsunari. Github. *MCL - A portable and fast pairing-based cryptography library.* (2011). Accessed: Jan. 19, 2024. [Online]. Available: https://github.com/herumi/mcl

[31] Information Security laboratory Okayama University. Github. *ELiPS - Efficient Library for Pairing Systems.* (2019). Accessed: Jan. 19, 2024. [Online]. Available: https://github.com/ISecOkayamaUniv/ELiPS