B.Tech Project Report

"Novel Construction of Comparable Attribute-Based Encryption: 0-Encoding and 1-Encoding Approach"

Prateek Kumar Singh, 20CS01003

Under the supervision of

Dr. Padmalochan Bera

School of Electrical Sciences

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR ,

December 2023

1. Introduction

In an era dominated by unprecedented volumes of digital data and an ever-expanding threat landscape, the imperative to secure sensitive information has never been more critical. Traditional encryption methods, while effective in many scenarios, often fall short when it comes to managing access to data in complex, dynamic environments. In response to these challenges, Attribute-Based Encryption (ABE) emerges as a revolutionary paradigm, offering a nuanced and flexible approach to data protection.

ABE represents a departure from conventional encryption models by enabling access control based on attributes rather than traditional cryptographic keys or passwords. This innovative cryptographic technique provides a granular and adaptive means of securing information, aligning more closely with the intricacies of modern data-sharing scenarios. There are several algorithms for ABE like KP-ABE and CP-ABE.

In several distributed systems a user should only be able to access data if a user possesses a certain set of credentials or attributes. Currently, the only method for enforcing such policies is to employ a trusted server to store the data and mediate access control. However, if any server storing the data is compromised, then the confidentiality of the data will be compromised. Ciphertext-policy attribute-based encryption presents a system for realizing complex access control on encrypted data that we call. By using these techniques encrypted data can be kept confidential even if the storage server is untrusted; moreover, these methods are secure against collusion attacks. Previous attribute-based encryption systems used attributes to describe the

encrypted data and built policies into user's keys; while in the system attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt. Thus, these methods are conceptually closer to traditional access control methods such as role-based access control .

Ciphertext-policy attribute-based encryption (CP-ABE) to address this problem, and give the first construction of such a scheme. In the system,a user’s private key will be associated with an arbitrary number of attributes expressed as strings. On the other hand, when a party encrypts a message in our system, they specify an associated access structure over attributes. A user will only be able to decrypt a ciphertext if that user’s attributes pass through the ciphetext’s access structure. At a mathematical level, access structures in our system are described by a monotonic “access tree”, where nodes of the access structure are composed of threshold gates and the leave describe attributes. We note that AND gates can be constructed as n-of-n threshold gates and OR gates as 1-of-n threshold gates. Furthermore, we can handle more complex access controls such as numeric ranges by converting them to small access trees

ENCRYPTION

The process of converting the original representation of the information, known as plaintext into an alternative form known as cipher-text. Modern encryption schemes use the concepts of public-key and symmetric-key.

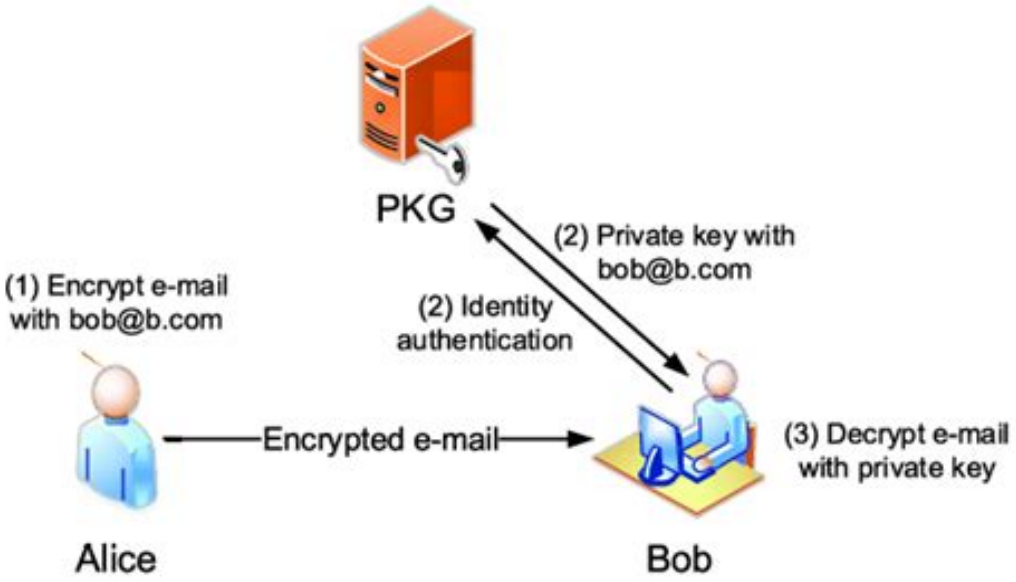
In symmetric-key schemes, the encryption and decryption keys are the same. Communicating parties must have the same key in order to achieve secure communication. In public-key encryption schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read.

Fine-grained Access Control

Fine-grained access control systems facilitate granting differential access rights to a set of users and allow flexibility in specifying the access rights of individual users. Identity-Based Encryption (IBE) is a cryptographic scheme that is related to the concepts of fine-grained access control.

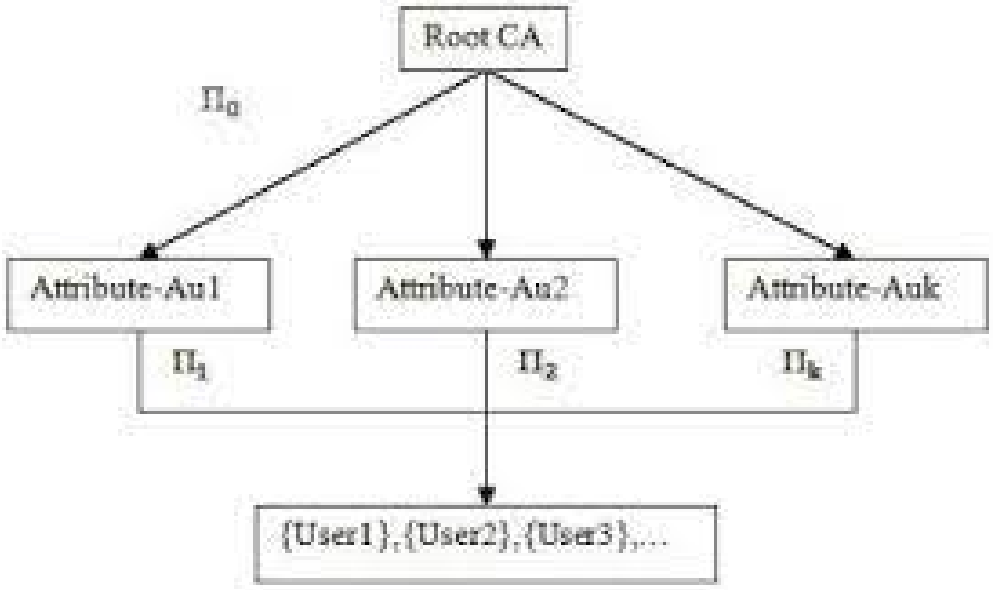
Identity Based Encryption

Identity based encryption is a type of public-key encryption in which the public key of a user is some unique information about the identity of the user (e.g. a user's email address). This means that a sender who has access to the public parameters of the system can encrypt a message. The receiver obtains its decryption key from a central authority, which needs to be trusted as it generates secret keys for every user.



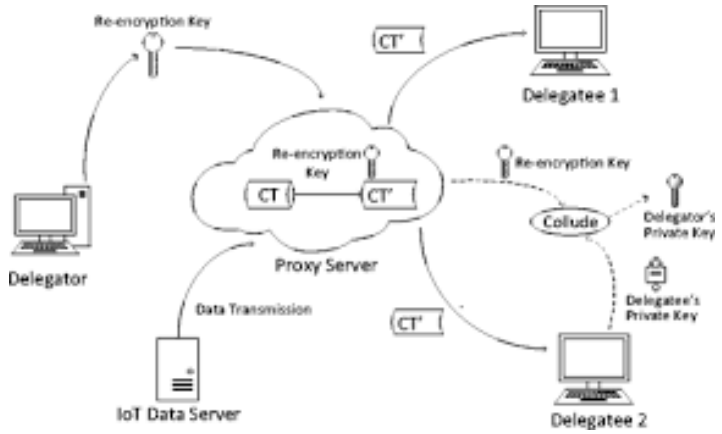
ATTRIBUTE BASED ENCRYPTION

Attribute-based encryption is probably a generalization of identity-based encryption which enables fine grained access control of encrypted data using authorisation policies. The secret key of a user and the ciphertext are dependent upon attributes (e.g. their email address, the country in which they live, or the kind of subscription they have). In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext.Researchers have further proposed attribute-based encryption with multiple authorities who jointly generate users' private keys. There are mainly two types of attribute-based encryption schemes: Key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE).



COLLUSION RESISTANCE

An important property which has to be achieved by both, CP-ABE and KP-ABE is called collusion resistance. If multiple users try to access the encrypted data, they should only be able to decrypt a ciphertext if at least one of the users could decrypt it on their own. Both CP-ABE and KP-ABE provide Collusion Resistance.



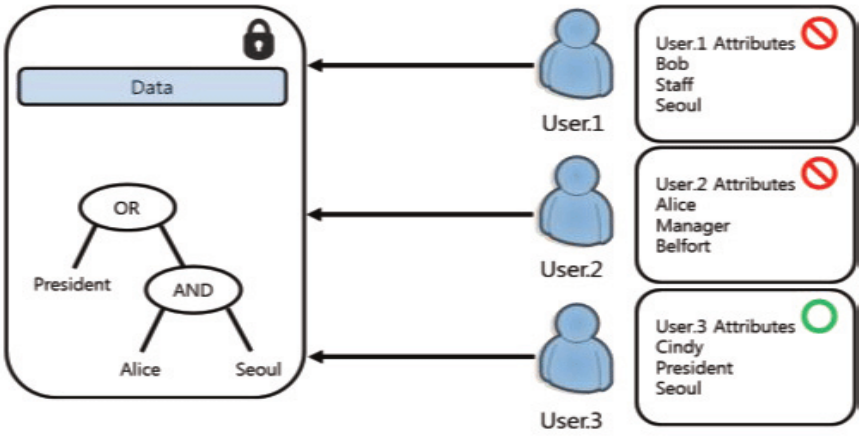
KEY-POLICY ATTRIBUTE-BASED ENCRYPTION

In KP-ABE, users' secret keys are generated based on an access tree that defines the privileges scope of the concerned user. Then Ciphertexts are associated with sets of descriptive attributes, and users' keys are associated with policies. In key-policy ABE, the encryptor exerts no control over who has access to the data she encrypts, except by her choice of descriptive attributes for the data. Rather, she must trust that the key-issuer issues the appropriate keys to grant or deny access to the appropriate users. In other words, the "intelligence" is assumed to be with the key issuer, and not the encryptor. Which is not the case with the CP-ABE.

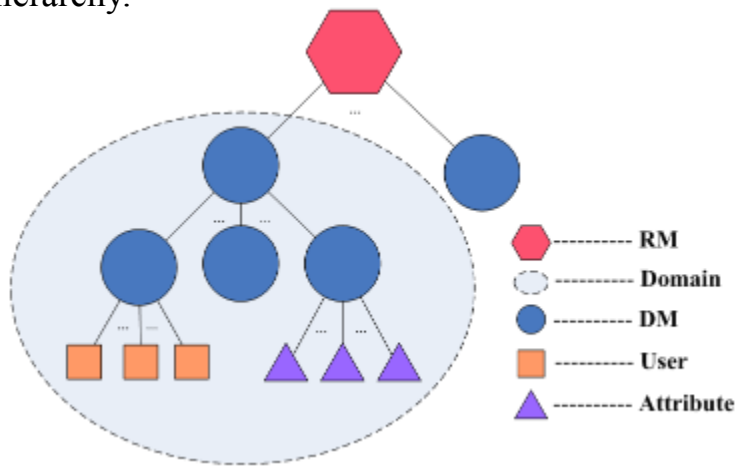
ACCESS POLICIES

Access policies refer to rules that govern the permissions and restrictions placed on entities attempting to access cryptographic resources. Here are some types of access policies commonly used:

- AND-Gate Policy: This policy requires that a user possesses all specified attributes for access.
- OR-Gate Policy: An OR-Gate policy allows access if a user possesses at least one of the specified attributes.



- Hierarchical Policy: In a hierarchical policy, attributes are organised in a hierarchical structure. Users can be granted access based on their position within the hierarchy.

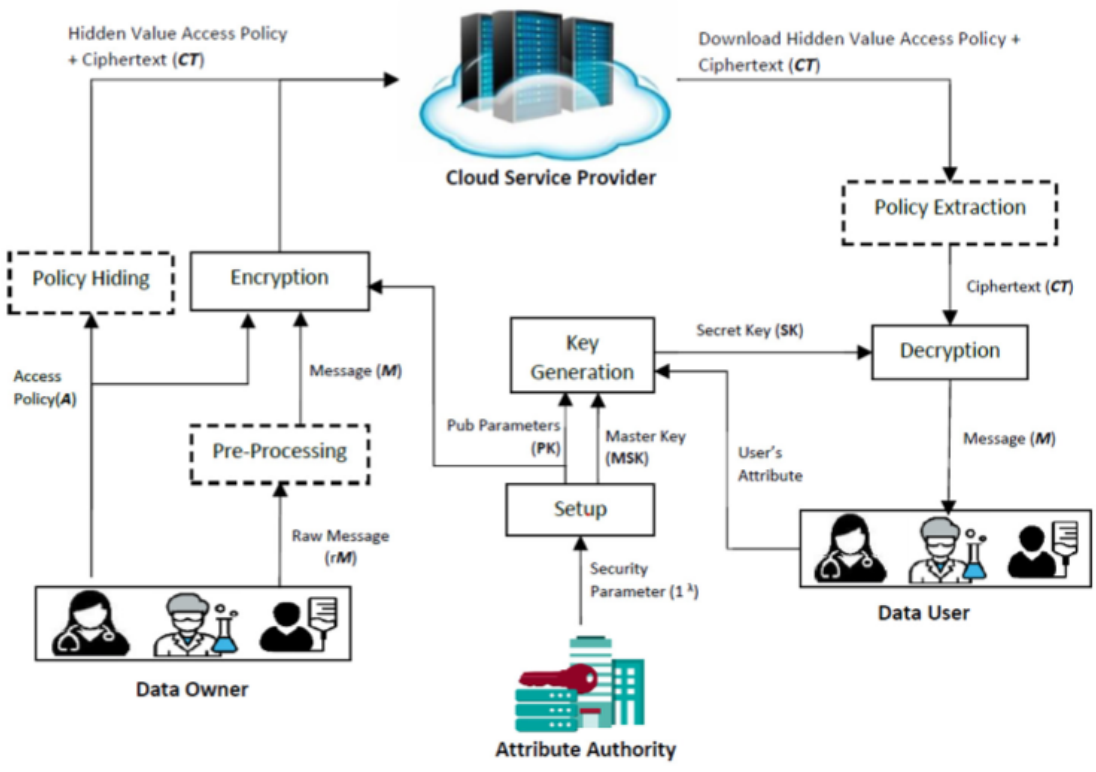


2. CipherText policy attribute based encryption

In ciphertext-policy attribute-based encryption (CP-ABE) ,a user's private key will be associated with an arbitrary number of attributes expressed as strings,from the universe of attributes available.

On the other hand, when a party encrypts a message in our system, they specify an associated access structure over attributes. A user will only be able to decrypt a ciphertext if that user's attributes pass through the ciphertext's access structure.

At a mathematical level, access structures in our system are described by a monotonic "access tree", where nodes of the access structure are composed of threshold gates and the leaves describe attributes. We note that AND gates can be constructed as n-of-n threshold gates and OR gates as 1-of-n threshold gates. Furthermore, we have implemented more complex access controls such as numeric ranges by converting them to smaller access trees using 0-1 Encoding for comparison.



A ciphertext-policy attribute based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt. In addition, we allow for the option of a fifth algorithm Delegate.

1] Setup.

The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

2] Encrypt(PK,M, A).

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

3] Key Generation(MK,S).

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

4] Decrypt(PK, CT, SK).

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

5] Delegate(SK, $\tilde{S}$).

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

3. The Problem

In the current implementation for the CP-ABE include only AND and OR gates which allows us to define the access policy based on them for example : ((“Public Corruption Office” AND (“Knoxville” OR “San Francisco”)) OR(management-level > 5) OR “Name: CharlieEppes”).

However, AND gate alone cannot designate intended users well in many cases. Thus the schemes supporting threshold gates have been attracting much more attention. An encryption and decryption algorithm with fewer participating attributes and less complex policies is preferred. The cost increases linearly with the attribute number. However, the value comparison in access policy should expand the attribute number of the structure. Thus, an inefficient mechanism may affect the feasibility of the scheme.

The comparison operation between a user’s and a file’s attributes only includes “=”.These attributes are numerical information,and will probably be used in comparison. For instance, a user may be assigned a key embedded with access policy as: “(Distance < 1000 miles) AND (Date > May 1st)” For example,

“Distance < 1000 miles” can be represented as a formula like (“Distance = 999 miles” _ “Distance = 998 miles” _ ... _“Distance = 0 miles”), but the overhead increases linearly with the growth of attribute’s value space, which will become a performance bottleneck of the system. This scheme divided such numerical attribute into pieces in units of bits as several sub-attributes to solve this problem. However, the mechanism to design a numeric-comparison policy is too complex, and the most essential problem is that the additional overhead is still relatively high in both Space and Time.

DATA STRUCTURE

In this paper by (Kaiping Xue, Senior Member, IEEE,), they proposed a new scheme to enable ABE to implement comparable attributes, called Comparable Attribute-based Encryption (CABE). In CABE, we use a unique way to generate and manage sub-attributes for comparable attributes, so as to make a solution for addressing the above issue in an efficient way, in terms of both storage overhead and computation overhead. Our solution in dealing with sub-attributes are based on a special notion called 0-encoding and 1-encoding. The main contributions of this paper can be summarized as follows:

1) We propose an efficient method based on a special concept 0-encoding and 1-encoding, so that the attributes can be used in arbitrary comparison, which is suitable for ABE system;

2) A lightweight and efficient CABE construction is proposed. This construction halves the expanded storage overhead in average compared with related schemes, and significantly decreases the computation overhead in encryption and decryption from O(log n) to O(1) (N denotes the value space of the attribute dimension).

Definition of 0-Encoding and 1-Encoding

Our system uses the concept of two special encodings, 0-encoding and 1-encoding. Let s=snsn-1….s1 ∈ {0,1}n be an n-length binary string of a value for a certain attribute dimension.

The 0-encoding of s is defined as a set S0s such that

$$S_s^0 = \{s_n s_{n-1} \dots s_{i+1} 1 | s_i = 0, 1 \leq i \leq n\}.$$

The 1-encoding of s is the set S1s such that S1s

$$S_s^1 = \{s_n s_{n-1} \dots s_i | s_i = 1, 1 \leq i \leq n\}.$$

Intuitively, 1-encoding of s is the set of all its odd prefix substrings, and the 0-encoding is the set of all of its modified even prefix substrings, where the least significant bit is flipped from “0” to “1”. The size of set S0s equals to the number of characters “0” in string s, and meanwhile the size of S1s equals to the number of “1”.

Compared with the value space of n-length binary string: N=2n, both S_{1s} and S_{0s} have at most log N elements. To compare two integers x and y in the form of n-length binary string, we encode x into 1-encoding S1x and y into 0-encoding S0y . We make the judgment that x>y if and only if there’s an element in both S1x and S0y . A formula to express this theorem is as

$$x > y \Longleftrightarrow S_x^1 \bigcap S_y^0 \neq \varnothing.$$

0-Encoding and 1-Encoding of 11 and 6

	1-encoding	0-encoding
$x=1011_2$	1 101 1011	11
$y=0110_2$	01 011	1 0111

Storing 0-Encoding and 1-Encoding

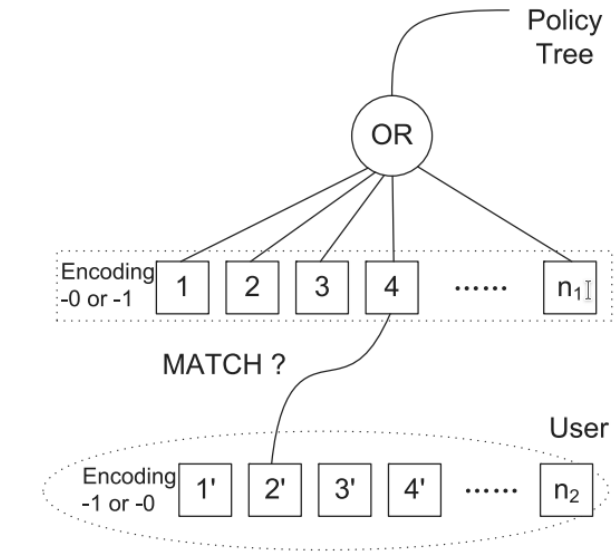
The object we deal with is such an attribute that is not an exact value, but stands as a range of continuous values, and may be matched in comparison in the ABE system, such as “Score > 75”, “Age < 25”. For one attribute field F, whose value space is N and minimum value is val_{min}, we can reduce the storage overhead to ½(logN) on average to make this kind of attribute suit CP-ABE construction with utilization of 0-encoding and 1-encoding. Our procedure is implemented as follows For a value x F, if its minimum value is not zero, compute

$x_m=x - val_{min}$;

otherwise, such operations can be skipped. For clarity of description, let $x_m=x$; in this case. If the length of the binary string xm is less than $\log_2 N$, use 0 to fill at high bits. Then what the program deals with is a new $\log_2 N$-long binary string x_m. We encode x_m into 0-encoding S_0

x_m and 1-encoding S_1 x_m with the rule described in Section 3.1. For a certain access structure, if the access policy needs the corresponding attribute F to satisfy that F > x, an attribute set $Set_{c0}(F,x)$ can be designed as

$$Set_{c0}(\mathcal{F}, x) = \left\{ (\mathcal{F} || \text{“} > x \text{”} || c) | c \in S^0_{x_m} \right\}.$$



Model of comparison and matching for comparable attribute.

PROPOSED EFFICIENT RANGE QUERY SOLUTION

Since we already exposed the mathematical construct on how to reduce the numerical comparison to simple some binary AND and OR operations we will now propose our solution for efficient range query. Some lemmas must be mentioned before proposing our solution :

Lemma 1 : Encode a into it's 0-Encoding set E_0 and b into it's 1-encoding set E_1 , we have b > a if and only if E_1 and E_0 have some elements in common.

From this lemma we can derive the theorem that the common element will be a single element.

A further observation that can be made from this is that if there is no common element between the 2 sets then b<=a.

Now in our solution we decided to perform some operations on the access policy with the range comparisons such that the intersection of the 2 0-Encoding and 1-Encoding set can be left up to the access policy tree itself and it will perform the intersection on the elements in 2 sets using the binary operations. This is done using the logic that say we want to find the intersection between 2 sets say set A and set B then if say we have 1 encoding of set A , the individual elements of 1 - encoding of the set can be taken in the form A1 OR A2 OR A3 OR A4 which are nothing but the individual elements of the 1-encoding.

Now say we get an actual value to compare then we can calculate it's 0-encoding and push that itself in the access tree and tell our access tree to perform the intersection function by doing comparison of individual elements thus resulting in intersection of the 2 sets of 1- Encoding of A and 0-Encoding of B and if access tree returns 1 that is true then it has found an element in intersection and thus we have successfully calculated that b < a.

Similarly we can swap the encoding schemes for integers and get the result
" x > value "
Now to implement the negation comparison for an attribute like " x!= value " we decided to do a deep dive in boolean logic along with set theory and for the given conditions we came up with the solution that if " x < value or x > value " returns a 0 value that means x != value . Proof for this lies in the fact that if x == value then x < value returns None and x > value returns None and if we take None OR None we get None aka 0 aka False. Now any number except value be either > value or < value where the condition " x < value or x > value " will return true hence the negation condition can be implemented successfully. So we were able to

implement range based comparison by combing the 2 conditions like “x < a and x > b “ such that “x belongs to (b,a)” and “ x<a or x >a ” implying “ x!=a ”.

Hence we were able to efficiently implement range based comparison in the access tree compared to previously known inefficient method which basically involved iterating through all the numbers in a given range (1,n) such that it performed equality check on all individual numbers in the given range thus becoming increasingly inefficient when the range of numbers became very big as inherent complexity of that is O(n) in both time as well as more complexity in space due to overhead of maintaining the access structure as well.

In our proposed solution and implementation we have cut down the time complexity from O (N) to O (log N) and space complexity to O (log N) as well, which we will demonstrate via some benchmarking graphs that we calculated on our code.

To do this we have defined some auxiliary functions over existing charm crypto library whose screen shots have been attached below :

Def encode_number : is a function that returns the 0-Encoding and 1-Encoding of a number which is used further to perform the binary comparison for 2 numbers.

Def modify_access_policy : is a function that takes input the access policy with comparable range comparisons and performs the required replacements with the given 0 and 1 attribute sets such that the binary comparison can be performed by the access structure in our tree.

```python
def encode_number(x):
    # Convert the integer to a binary string
    binary_string = bin(x)[2:]

    # Pad the binary string with leading zeros if needed
    # standardize all binary strings to be of length 32
    binary_string = binary_string.zfill(32)
    # Initialize 0-encoding and 1-encoding sets
    S0_x = set()
    S1_x = set()

    # Iterate over the binary string and populate the sets
    for i in range(len(binary_string)):
        prefix = binary_string[:i+1]
        if prefix[-1] == '0':
            # Flip the least significant bit when adding to S0_x
            S0_x.add(prefix[:-1] + '1')
        else:
            S1_x.add(prefix)
    # convert the binary to decimal
    # sort S0_x and S1_x by length of the binary string
    S0_x = sorted(S0_x, key=lambda x: len(x), reverse=True)
    S1_x = sorted(S1_x, key=lambda x: len(x), reverse=True)
    return S0_x, S1_x
```

```python
def modify_access_policy(access_policy):
    access_policy = modify_not_equal_conditions(access_policy)
    pattern = re.compile(r'(\w+)\s*([<>])\s*(\d+)')
    matches = pattern.findall(access_policy)
    for match in matches:
        identifier, operator, number = match
        access_policy = access_policy.replace(
            f'{identifier} {operator} {number}', f'({identifier} {operator} {number})')
    matches = pattern.findall(access_policy)
    for match in matches:
        identifier, operator, number = match
        S0_x, S1_x = encode_number(int(number))
        if operator == '<':
            new_condition = ' OR '.join(f'{identifier}{"!!"}{x}' for x in S1_x)
        elif operator == '>':  # operator == '>'
            new_condition = ' OR '.join(f'{identifier}{"@@"}{x}' for x in S0_x)
        access_policy = access_policy.replace(
            f'({identifier} {operator} {number})', f'({new_condition})')
    pattern = re.compile(r'(\w+) = (\w+)')
    modified_policy = access_policy
    for match in pattern.findall(access_policy):
        old_expression = f'{match[0]} = {match[1]}'
        new_expression = f'({match[0]}$${match[1]})'
        new_expressionx = new_expression.upper()
        modified_policy = modified_policy.replace(
            old_expression, new_expressionx)
    return modified_policy
```

These 2 functions are primarily used for the calculation of 0 and 1 encoding of the numbers and further values are provided to access tree for comparison. Now we will observe the efficiencies of our code compared to the previous implementations of CPABE for range based comparisons :

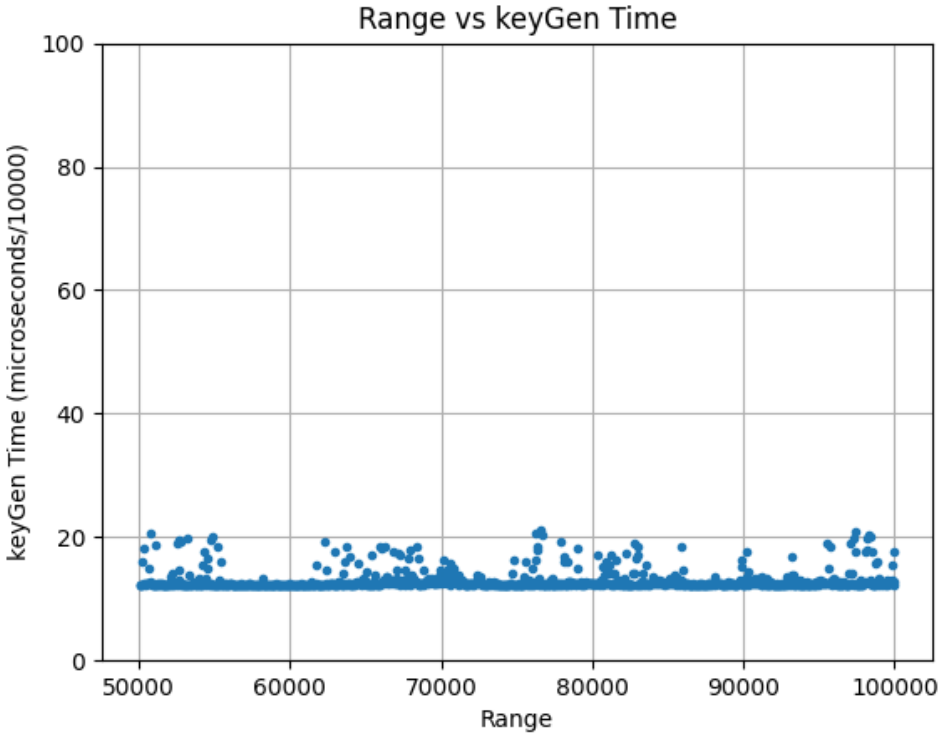
4. Result and analysis

3 kinds of graphs have been primarily used to evaluate the performance of our proposed algorithm which are

1. Key Generation Time : This represents the amount of time the algorithm takes to generate the public key and the secret key based on the access policy size .
2. Encryption Time : This represents the amount of time it takes to encrypt a given generic text file which can be examined while looking at our code .
3. Decryption Time : This represents the amount of time it takes to decrypt a given encrypted document so as to get the original information.

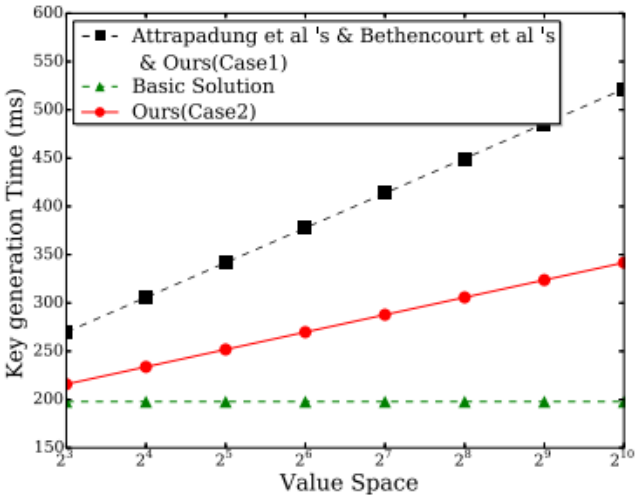
Theoretical calculations and graphs also have been provided along with our own benchmarks so as to show case the efficiencies of our algorithm. The most important graph can be the key generation symmetric and decryption time vs the actual range because our whole solution is about optimizing the extremely big numerical ranges compared to the generic implementation of CP-ABE which are extremely slow and memory hogging for moderately big values of numbers. Now

lets' observe the graphs obtained for extremely large value of ranges which were implemented in the access policy of the form " x < range1 and x > range2 ".

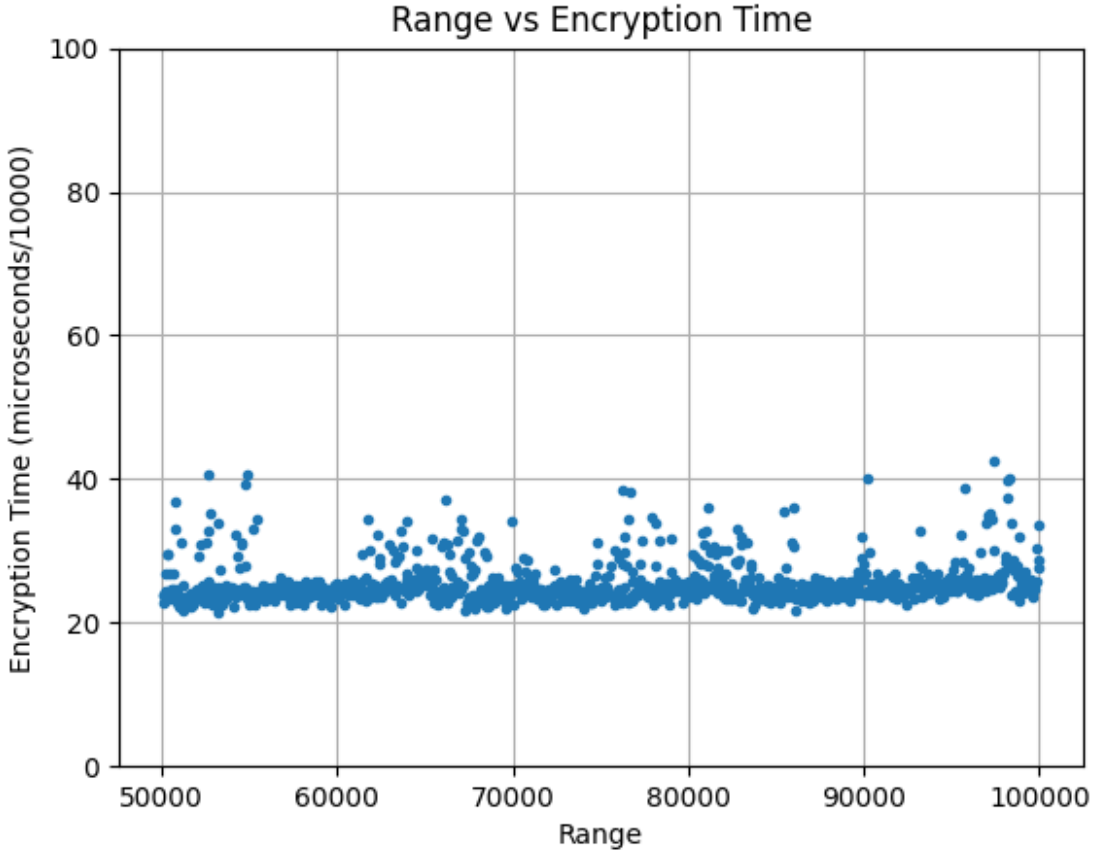


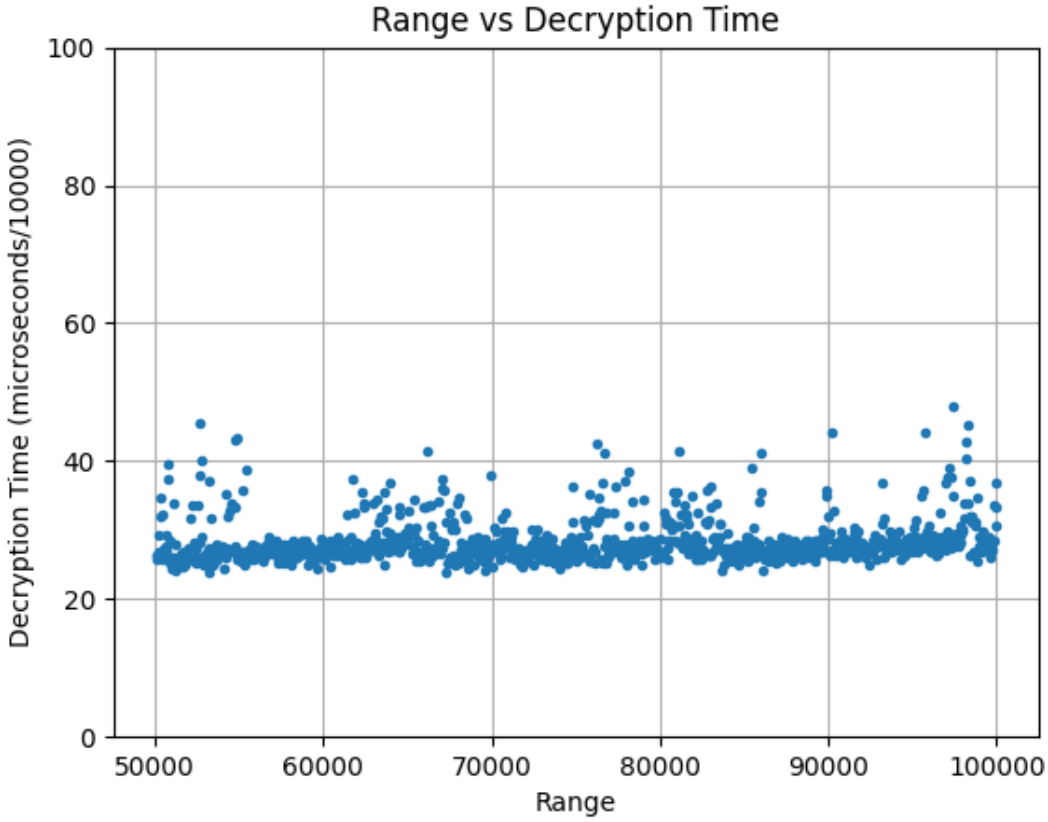
An extremely slow growing almost constant O (1) complexity is observed in the time consumption for the keyGen algorithm which proves that our algorithm is way faster than the generic implementation of range queries for CP-ABE which is O(N) in nature which blows up very quickly in terms of memory as well as time required for calculations.

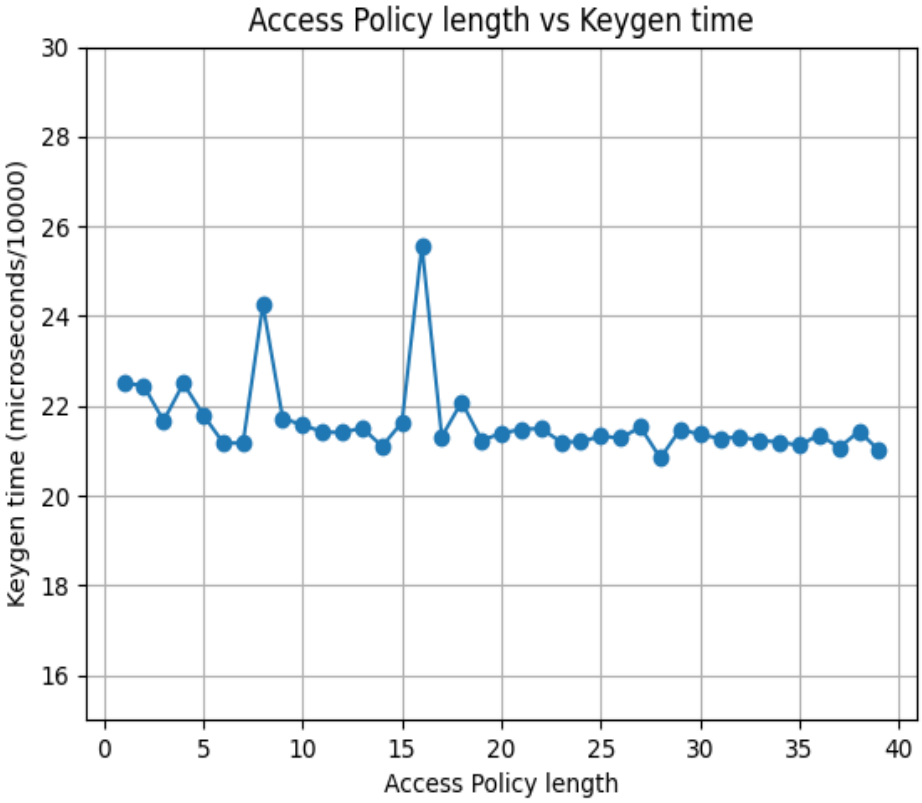
This almost constant time complexity is in line with the theoretical calculations that were proved in the paper that was used for reference.



Similar graphs are observed while calculating the encryption and decryption time Which are mentioned below

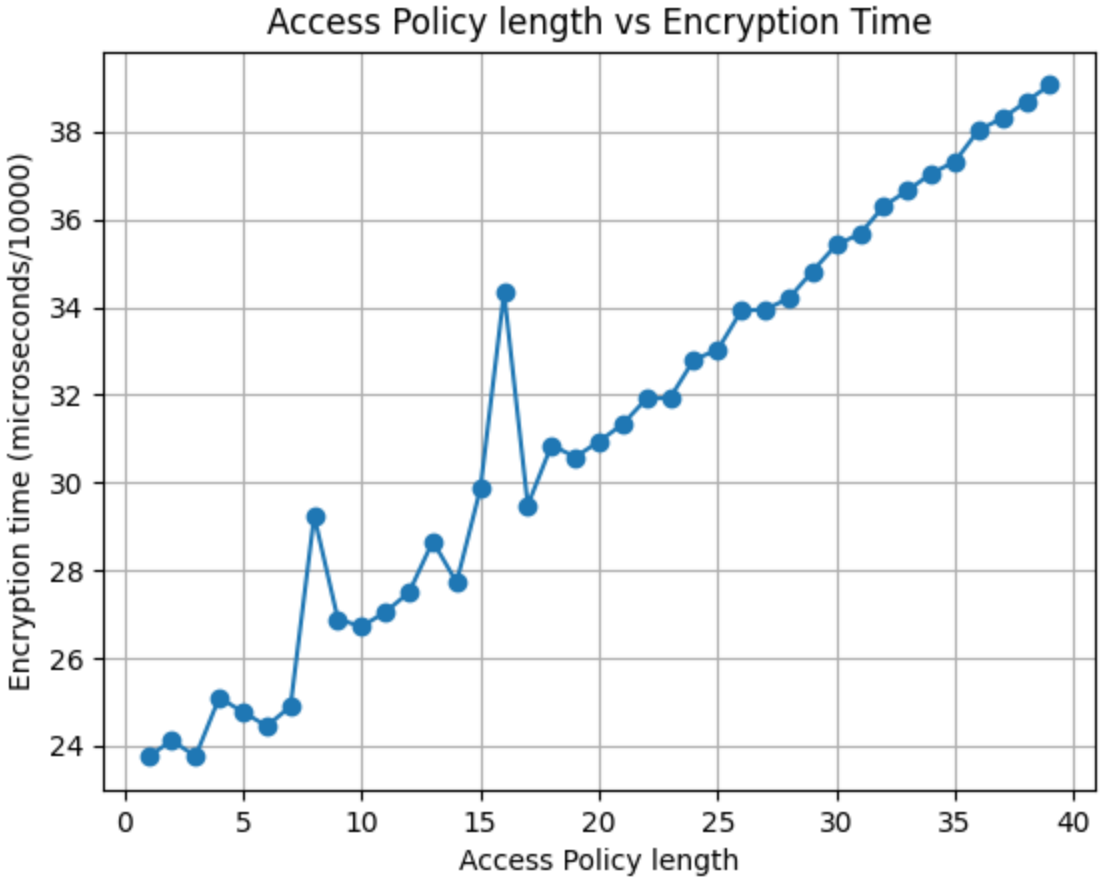


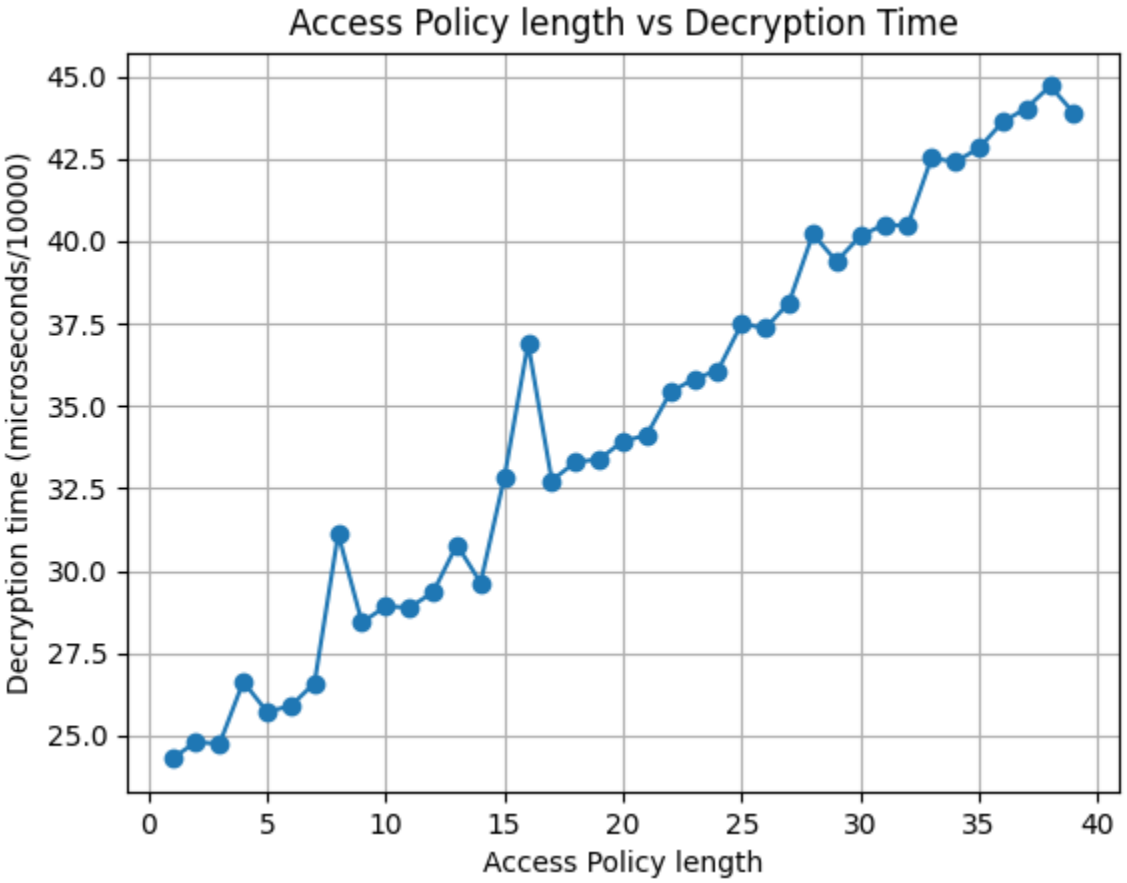




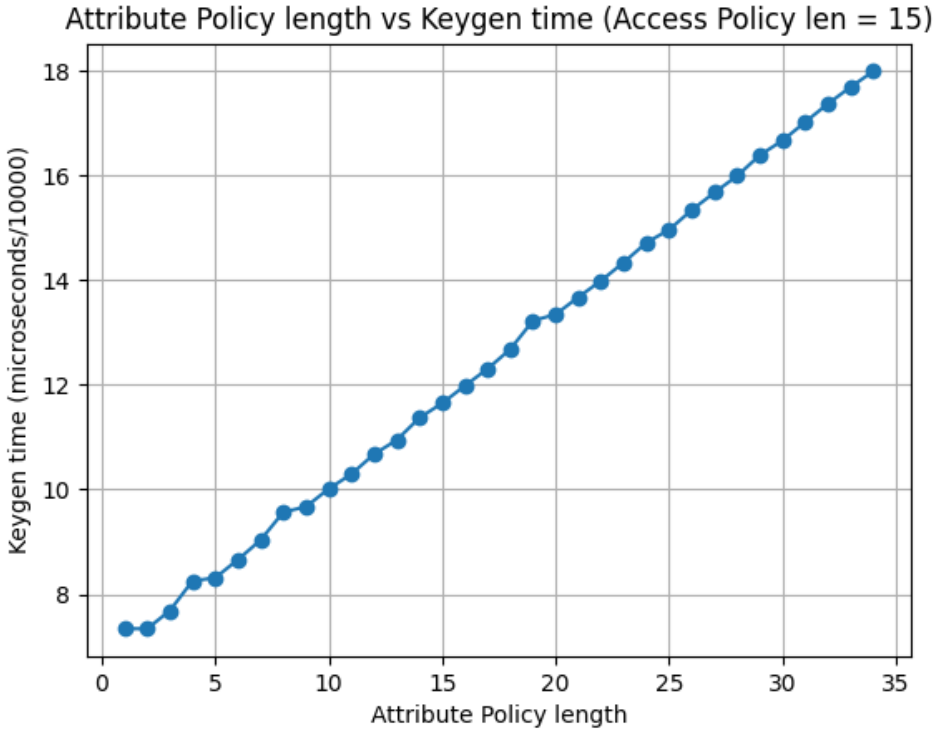
We observe that the time taken for Key Generation remains pretty much constant irrespective of the access policy length which is in line with our assumption that irrespective of the fact that how big the number of attributes get the growth is extremely slow approximately log (n) or barely constant.

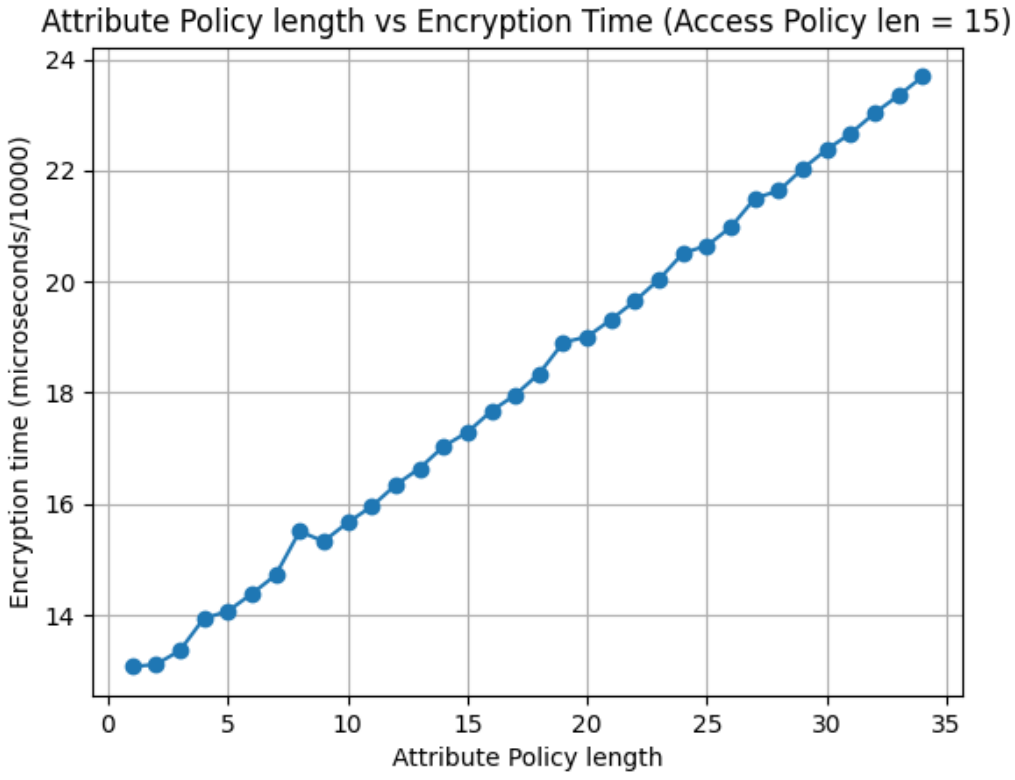
Similar graphs can be drawn for encryption and decryption time which can be used to draw some conclusions about the various optimisation techniques that can be used in the decrypting process which can be observed in the next page.

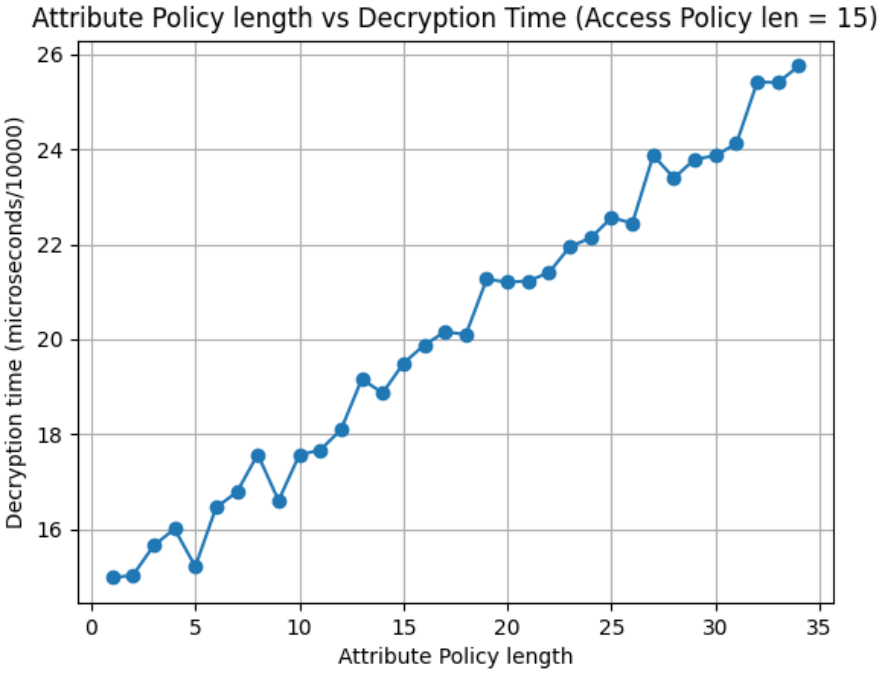

Access Policy length vs Encryption Time
Encryption time (microseconds/10000)
Access Policy length
24
26
28
30
32
34
36
38
0
5
10
15
20
25
30
35
40


Access Policy length vs Decryption Time
Decryption time (microseconds/10000)
Access Policy length
25.0
27.5
30.0
32.5
35.0
37.5
40.0
42.5
45.0
0
5
10
15
20
25
30
35
40

In case of attribute length as a parameter which represents the number of parameters present in the user who is trying to decrypt the encrypted file and linearly increasing but stable relationship is observed between the key generation time and number of parameters in Attribute for a given user.







Some uniformity is observed in the encryption algorithm where as some non linearity can be observed in the decryption algorithm which is due to varying level of optimisations being performed with the given access structure and given attribute properties array for the user attempting to decrypt the file.

References :

1. CABE: A New Comparable Attribute-Based Encryption Construction with 0-Encoding and 1-Encoding
   Kaiping Xue, Senior Member, IEEE, Jianan Hong, Yingjie Xue, David S. L. Wei, Senior Member, IEEE, Nenghai Yu, and Peilin Hong
2. Efficient Encrypted Range Query on Cloud Platforms
   PING YU, WEI NI, REN PING LIU, ZHAOXIN ZHANG , HUA ZHANG and QIAOYAN WEN,
3. Ciphertext-Policy Attribute-Based Encryption
   John Bethencourt , Amit Sahai , Brent Waters