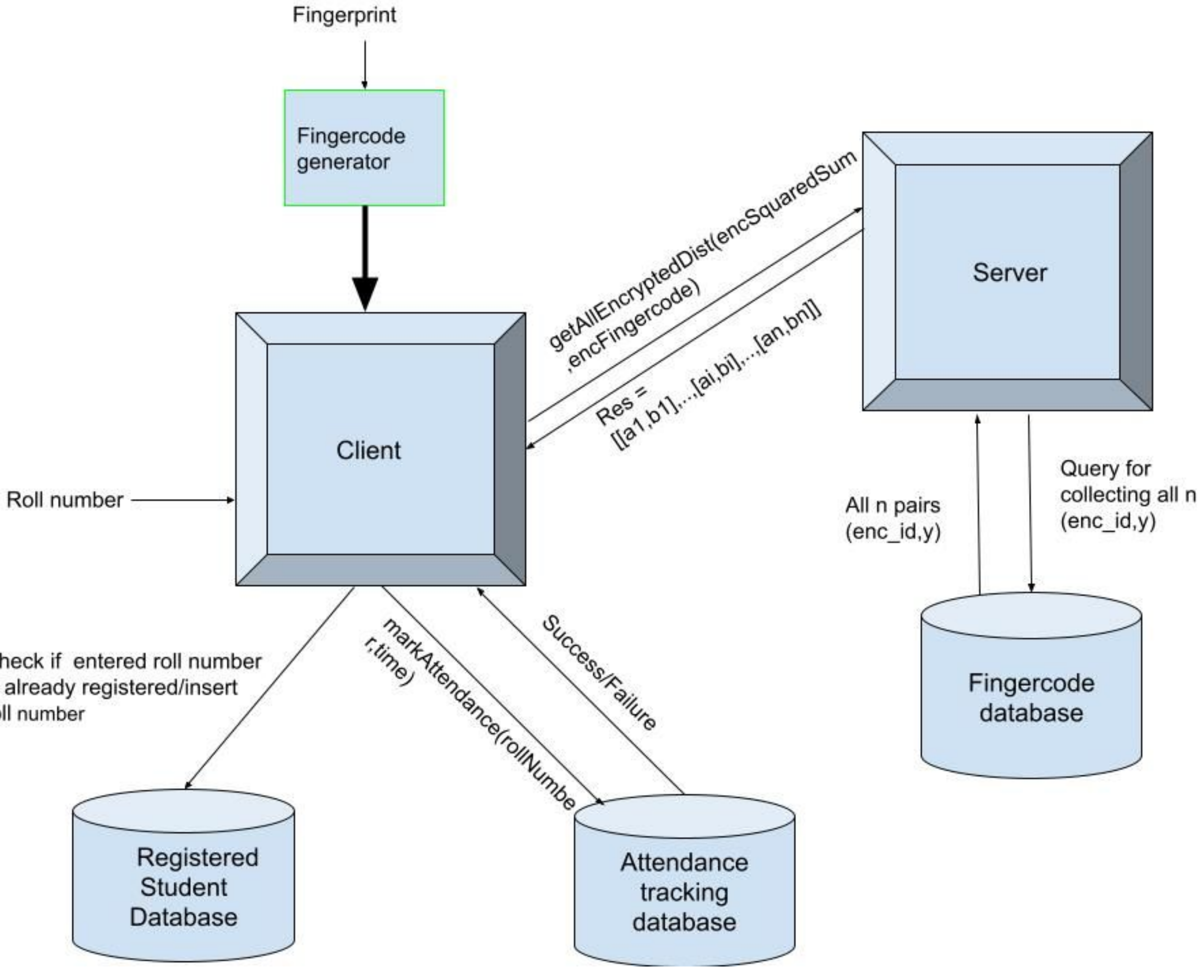
Fingerprint
Fingercode generator
Client
Server
Roll number
getAllEncryptedDist(encSquaredSum ,encFingercode)
Res = [[a1,b1],..[ai,bi],..[an,bn]]
Query for collecting all n (enc_id,y)
All n pairs (enc_id,y)
Fingercode database
Check if entered roll number is already registered/insert roll number
markAttendance(rollNumber,time)
Success/Failure
Registered Student Database
Attendance tracking database

## ASSUMPTIONS

1. The algorithm to convert a fingerprint into a fingercode is available and the mapping from fingerprint to fingercode is injective.
2. Since we don't have access to an actual fingerprint scanner, it is assumed that a random list consisting of positive integers represents a fingercode of each student and for marking attendance the fingerprint is read and is converted to fingercode using some algorithm.
3. Generated fingercode is directly passed to the client machine through a secure channel.
4. Roll Number is a positive integer of 5 digits and the fingercode is a random list containing 6 positive integers.
5. Reconstructing the fingerprint from a given fingercode is very hard.

## DATABASE

1. The schema of **Fingercode Database** consist of 2 columns *DATA1* i.e. encrypted roll number and *DATA2* i.e. fingercode for every student (i.e all the students are registered with their fingercodes).
2. The schema of **AttendanceTracking Database** consists of 2 columns *ROLLNUMBER and TIMESTAMP*. Initially, the database is empty.
3. The schema of **RegisteredStudent Database** consists of 1 column *ROLLNUMBER* i.e roll numbers of registered students in plaintext form**.** Initially, the database is empty.

## WORKING OF ATTENDANCE MARKING SYSTEM

1. For the registration part, when the roll number is entered through some interface provided by Local Server, it is checked whether this entered roll number is already registered in the RegisteredStudent Database. If the roll number is already registered, an error is reported otherwise a fingerprint is read and converted into fingercode, then the entered roll number is inserted into the RegisteredStudent database and encrypted roll number along with the fingercode is inserted into the Fingercode Database.
2. For marking attendance the client server will first read rollNumber and fingercode of the user. Then isPresent(defined below) function is called for checking the existence of such a user.
3. If isPresent returns True then client will mark the attendance of the user with id “rollNumber” using interface markAttendance(rollNumber,timeStamp) by inserting a new row with ROLLNUMBER = rollNumber and TIMESTAMP = timeStamp in attendanceTrack database and returns True if attendance marking is successful otherwise for any internal error it returns False.
4. If isPresent returns False Local Server will not mark the attendance and display an error message.

### Function - GenerateFingerCode

Input: Fingerprint grayscale image<br>
Output: Fingercode - a list of 6 integers.

GenerateFingerCode(fingerprintImage):

```
1. Select a reference point on the image.
2. Mark a circular region of interest around the reference point.
3. Divide this region into sectors.
4. The average absolute deviation from the mean of gray values in individual sectors is
   calculated to get the fingercode vector.
```

### Function - isPresent

Input: rollNumber and Fingercode = [a1,a2,...,ak]

Output: Boolean value.

isPresent(rollNumber,fingercode):

```
1. encFingercode = [encrypt(a1),encrypt(a2),...,encrypt(ak)] using paillier cryptosystem.
2. squaredSum = sum(ai * ai) for i = 1 to k
3. encSquaredSum = encrypt(squaredSum) using paillier cryptosystem.
4. Res = result of calling server function getAllEncryptedDist(encSquaredSum,encFingercode)
5. Threshold = 5
6. For (encRoll,encDist) in res:
    a. decryptDist = decrypt(encDist)
    b. If decryptDist < Threshold:
        i. decryptRollNo = decrypt(encRoll)
        ii. If decryptRollNo == rollNumber:
            1. Return True
7. Return False
```

### Function - getAllEncryptedDist

Input: encSquaredSum and encFingercode = [a1,a2,...,ak]
Output: List of pairs such that each pair contains encrypted roll number and encrypted squared euclidean distance between fingercode and the input fingercode.

```
getAllEncryptedDist(encSquaredSum, encFingercode):
        1. queryResult = list of (encryptedRollNo_stringType, fingercode) pairs from the database.
        2. res = empty list
        3. For (encryptedRollNo_stringType, fingercode) in queryResult :
            a. encryptedRollNo = int(encryptedRollNo_stringType)
            b. encryptedDistance = squaredEuclideanDistance(encFingercode , encSquaredSum
               ,fingercode)
            c. Append (encryptedRollNo ,encryptedDistance) to the res list.
        3. Return res.
```

### Function - squaredEuclideanDistance

Input: encFingercode = [z1 = encrypt(x1),z2 = encrypt(x2),...,zk = encrypt(xk)], encSquaredSum and Fingercode = [y1,y2,...,yk]
Output: encrypted squared euclidean distance [x1,x2,...,xk] and [y1,y2,...,yk].

```
squaredEuclideanDistance(encFingercode, encSquaredSum,fingercode):
        1. Res1 = encSquaredSum
        2. Res2 = product( $(zi)^{-2yi}$ ) for all i = 1 to k
        3. Res3 = encrypt(sum(yi * yi) for all i = 1 to k)
        4. Result = Res1 * Res2 * Res3 mod $N^2$.
        5. Return result
```

**Note : Partial Homomorphic Encryption properties are used while computing squared euclidean distance.**

### Security Analysis

- Since our system uses the paillier encryption scheme, therefore the security of our system depends on the security of the paillier encryption scheme.
- Even if the fingerprint database is compromised by a passive attacker, the attacker can only see the encrypted roll no and corresponding fingercode. And we have assumed that it is very hard to find out the fingerprint from the fingercode.