Merely.db is the database which contains 4 tables :

* keyStorage
* traineeRecord
* employeeList
* traineeProfile

the traineeRecord contains trainee’s encrypted record, which includes :

* QR record
* E51D3 (encrypted ID)
* EFT51D3 (encrypted finished tasks)
* ESR51D3 (encrypted self-rating)
* ED51D3 (encrypted date)
* Supervisor public key
* ESS51D3 (encrypted sign status)
* ESR151D3 (encrypted supervisor rating)
* EF51D3 (encrypted feedback)

The traineeProfile contains :

* Profile\_EF51D3 (encrypted ID)
* Profile\_EQR51D3 (encrypted QR record)

keyStorage contains :

* Ciphered trainee ID
* Trainee Public Key

\*\*\* Note : due to unable to put the long encrypted data in database, we decide to insert only the file name contains the encrypted data so user could find the data easily.

For example (E51D3.bin is the file contains Trainee’s encrypted ID)

Objective – supervisor wants to retrieve trainee’s record for making assessment.

What we have :

+ Trainee’s record is encrypted by trainee, the visible data are QR record and supervisor public key.

+ Trainee’s profile is completely encrypted by trainee.

+ Trainee’s key storage, only his ID is encrypted by Caesar Cipher method, his public key is visible but can only be retrieved by querying his ID.

+ The employee table is visible to user, supervisor can retrieve trainee’s ID from there.

What to do :

+ Supervisor encrypts trainee ID with Caesar Cipher method then uses that ciphered ID to find trainee public key in keyStorage table.

+ Supervisor encrypts trainee ID with the retrieved trainee public key, then find trainee’s encrypted QR code in traineeProfile.

+ The QR code is sent to trainee for decrypting, then encrypted by Supervisor public key and sent back to him.

+ With the decrypted QR code, supervisor is now retrieved the encrypted trainee record.

+ Trainee will decrypt the record and encrypt with supervisor public key then send back to him.

+ Supervisor decrypts the record and starts the assessment stage.