In the lab we learned how to read content from an NFC tag and how to write content to them. We also learned in the lecture that we can lock different sectors of a MIFARE Classic card with different keys. For this homework you will make an access system using this knowledge and the MIFARE Classic fob (the smiley face fob).

**System Parameters**:

- One or two Python scripts
  - The script should read a text file with an employee's details and create an entry card for them.
  - The script should also be able to read the entry card and determine if the user has the right to access a door
  - You can have:
    - add_user.py and check_user.py (this does not have to be the exact name of the file)
    - or employee_system.py -a (to add user) and employee_system.py -c (to check user)
  - You can give the files any name, just add a comment in the file for the execution instructions.

**How the add user system works**:

- When a new employee joins the company you receive a text file with their name, the department and clearance level. The file takes the following format (sample files also on Moodle):

```
Danielle Morgan
Research and Development
3
```

- On receiving the file you read the contents and format a blank MIFARE Classic card for them in the following way:
  - Sector 1 (the second sector) should contain the name of the employee **[0.5 pts]**
    - Block 0 (1st block in this sector): EMPLOYEE:
    - Block 1: First name e.g. Danielle
    - Block 2: Last name e.g. Morgan
    - Block 3: Key A = 11 11 11 11 11 11, Key B = FF FF FF FF FF FF
  - Sector 2 (the 3rd sector) should contain the name of the department **[0.5 pts]**
    - Block 0 (1st block in this sector): DEPARTMENT:
    - Block 1 and 2: Department e.g. Research and Development
    - Block 3: Key A = 22 22 22 22 22 22, Key B = FF FF FF FF FF FF
  - Sector 3 (the 4th sector) should contain the clearance level **[0.5 pts]**
    - Block 0 (1st block in this sector): CLEARANCE:
    - Block 1: number e.g. 3
    - Block 3: Key A = 33 33 33 33 33 33, Key B = FF FF FF FF FF FF
  - Sector 4 (the 5th sector) will contain your authentication information **[0.5 pts]**
    - Block 0 (1st block in this sector): AUTHENTICATION:
    - Block 1: the UID of the card being formatted
    - Block 2: Secret code
    - Secret code is a sha256 hash of the employee name, department and clearance
    - Block 3: Key A = 44 44 44 44 44 44, Key B = FF FF FF FF FF FF
  - The access conditions for the blocks (only for the 4 sectors used) **[0.5 pts]**
    - Data Blocks: Read with Key A or B and write with Key B, increment and decrement not allowed
    - Sector Trailer – write Key A with Key B, read access bits with Key A or B, write access bits with key B and write Key B with Key B. Reading of Key A and B are not allowed
  - Create a file and add the user's secret code to this file. The name of the file should take the following format uid.txt. This will act as a database entry **[0.5 pts]**

For example, if the uid of the card is b0b1b2b3 then the file is named b0b1b2b3.txt and the contents (no spaces in the file name):

```
Secret code
```

**How the verify user system works**:

- Once a card is detected at the door, first confirm that the UID of the card matches the UID in sector 4, use Key A to read the sector. **[0.5 pts]**
  - If not "Access denied" else move to next step
- Check if there is a file in the database named [uid].txt (the uid of the card .txt) **[0.5 pts]**
  - If not "Access denied" else move to next step
- Read the name of the employee, department and clearance from the card and calculate the secret code. Compare the calculated secret code with the one on the card and the one in the [uid].txt file
  - Key A should be used to read the contents of the card **[1 pts]**
  - If they do not match "Access denied" else move to next step **[0.5 pts]**
- Check that the clearance level number of the employee is higher than (or equal to) the clearance level of the door being accessed **[0.5 pts]**
  - If not "Access denied", else "Door opens"
  - The access level for the door can be randomly generated when the script initiates

**Notes**:

- Ideally Key B would be unique for each card and/or sector but to make it easier for testing and future rewrites we will leave the value as FF FF FF FF FF FF.

**Secret code generation**:

A hash function is one that can be used to match data of random length to fixed-size values. This means that no matter the length of the input the output length will always be the same. This output is known as a hash.

We will use the sha256 algorithm to calculate our secret code or hash. This can be done using the hashlib library. We will only need 16 bytes as each MIFARE Classic block is only 16 bytes in length.

Any 16 bytes of the hash can be used, first 16, last 16 etc. but it has to be consistent so your check does not fail when verifying the user.

Documentation: https://docs.python.org/3/library/hashlib.html

import hashlib
secret = hashlib.sha3_256((name+department+clearance).encode()).hexdigest()[:16]

e.g.

```
secret = hashlib.sha3_256("Danielle MorganResearch and Development3".encode()).hexdigest()[:16]
```

***N.B.*** The name of the employee, department etc. from the file should be used here not the word "name", "department" etc.

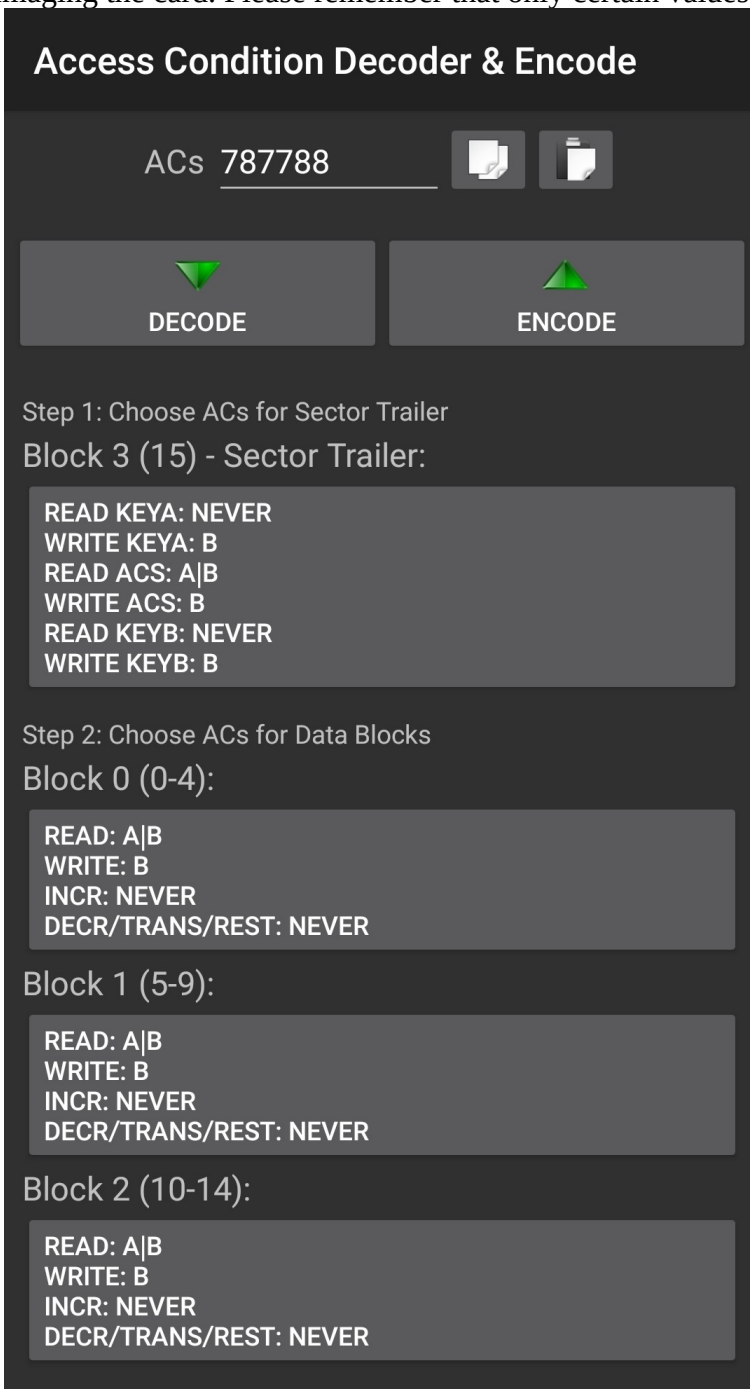
**Random Door clearance level generation:**

- `door_level = randrange(6)` can be used

**Access bits calculation:**

- Any value can be used for the 4th byte of the access bits (or the general purpose byte)
  - Example sector Block
  - 11 11 11 11 11 11 78 77 88 41 FF FF FF FF FF FF
  - or be creative maybe the first nibble specifies the length of the first name of the employee and the second nibble the length of the second or the department length
  - 11 11 11 11 11 11 78 77 88 86 FF FF FF FF FF FF

- The access conditions have been calculated for you (see image) to prevent incorrect access conditions damaging the card. Please remember that only certain values are allowed




**Hints:**

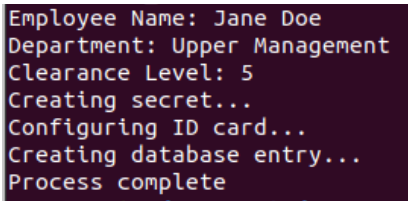
- Use the reset_card.py or reset_card2.py to erase the contents of the card
- Write the data blocks first then configure the sector trailer
- Print your content first, ensure the right lengths then write to the card

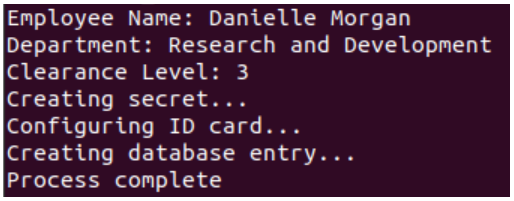
**Things you can assume (no need for checks):**

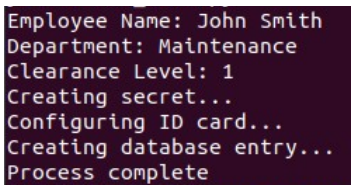
- The employee file will always be named new_employee.txt i.e. employee1.txt, employee2.txt does not exist
- There will only be one employee's details in the file
- Each field (name, department, clearance level) will be <= 32 bytes

- The employee will only have 2 names (first and last): each <= 16 bytes
- The employee file will be in the same folder as the python script
- The database file will be in the same folder as the python script
- The MIFARE card will always be at factory settings before adding user information i.e. A keys and B keys are all FFs (as such the fobs provided for testing currently have default settings – reset_card.py will also do this for you. Execute it twice if the Sector Trailer has been configured)
    - This means that Key B cannot be used to access data blocks (see page 36 in Lecture)

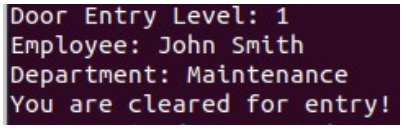
**Screenshots**
*Add User*



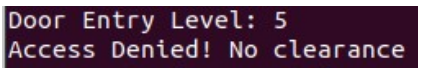



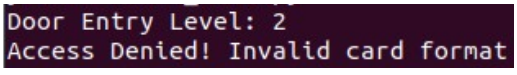

*Check User*
Access denied is a good encough message – these are more for testing purposes:
All checks pass:



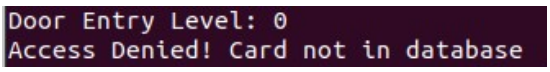

Failed Clearance Level:



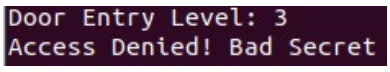

Failed UID check:

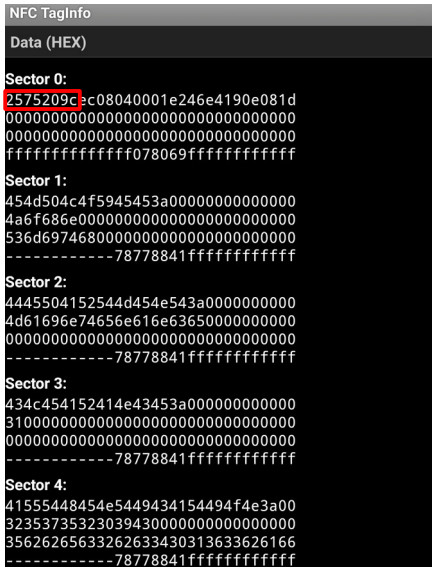



Failed file in database check:

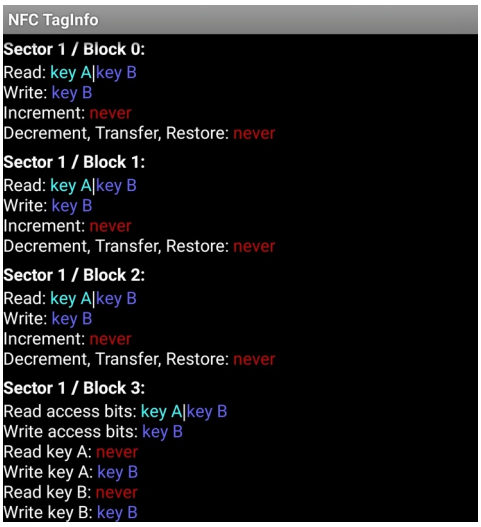


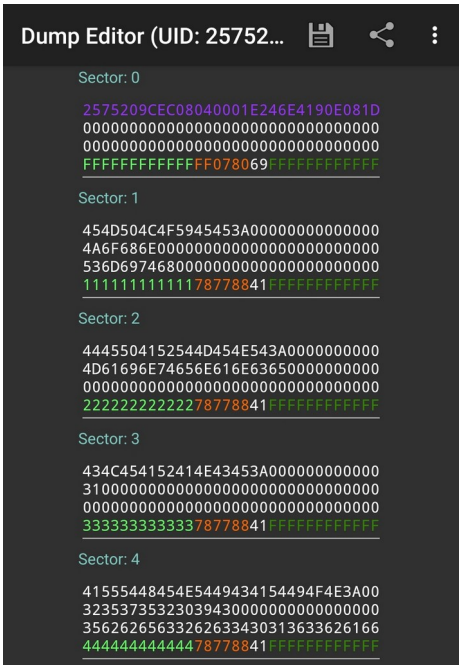

Failed the secret check:










**Things to consider about your system:**

1. Can an attacker create a valid clone to gain access? What would they need to do?
2. What form of security does comparing the UIDs provide?
3. Is 11 11 11 11 11 11 (22, 33, 44) a good key value?
4. If Key A was different for every employee card how would the system know which key was needed for reading?
5. Should Key B be the same value for every sector?
6. Should Key B be the same value for every card?
7. What access conditions (i.e. read Key A with Key B etc.) would be the most ideal for this system?
8. What form of security does the hash provide?
9. How can this system be improved?
10. How would you transfer this access card system to an Ultralight C card? Consider memory size and read/write locking capabilities.

**Due Date**: Check Moodle for date
**Late Submission Deadline**: Check Moodle for date (usually 1 week after due date)
**Submission Files**: .py files
**Location**: Moodle Homework 3 link

**Additional Notes**

- Template files are provided but you can ignore them and create your own file
- You can also use the sample code from the lab but use your own programming style e.g. if you prefer more white space or a different naming scheme for functions/variables
- Anything you are unsure about, please email/message me
- Please do not copy code directly from online sources (not that there would be anything specific to this assignment but…)
- Please do not copy code from other students. Asking for an explanation on how something works is fine, copy and pasting is not.
- Use the ACR1252 API Manual if you are unsure of the format for command APDUs
- The questions above do not have to be answered
- It is not as hard as you think or it seems :)