Assignment 5: Problem 1

System Design

System Design Overview

When a guest checks in or a staff member joins the hotel, they recieve a keycard with the data specified in the keycard configuration below. Then, when the guest uses the keycard, the lock will see it is not a staff member and proceed to validate that the guest checkout time has not passed and than the hash stored on the keycard is compatible (more on this below) to the hash stored for guests on the lock. Then, if everything is verified, access is granted to the room. For staff members, the lock will recognize that it is a staff keycard and proceed to check if the hash stored on the keycard is compatible to the hash stored for that specific staff member. Then, if everything is verified, access is granted to the room.

Encoder Configuration

Let's assume that the hotel is extremely popular and has maximum occupency of 600 guests a night with an average of 2 keycards per room. This comse out to a total of 216,000 keycards that are issued in a 6 month period. We will thus store 216,000 256-bit hashes for a total of 6912 KiB of hashes. These will be used in the chaining mechanism for guests explained in the design goal 1 section.

The final thing is that the encoder must be able to accept input; specify a guest's checkout date, whether the keycard it is encoding is a staff keycard, and if so, what the staff member's identifier is as well as the staff member's password.

Lock Configuration

The information needed on the lock in terms of guests is one 256-bit hash.

For staff, we know that there are at most 100 staff members for any given 6 month period. Instead of chaining, staff members just have a password, and the hash of that password is stored on the lock. The password will be a max of 256 bits. Therefore, the lock must store a total of 100 256-bit passwords for a total of 3.2 KiB of hashes. These will be used in the authentication protocol described below.

Therefore, total information stored on the lock is 3.232 KiB, which is below the 16 KiB capacity.

In terms of configuration, there is only one item needed from the encoder. Here is the protocol:

```
Abbreviations: E = Encoder , PP = Portable Programmer, L = Lock
1: E->PP: Largest hash in hash chain for the room
2: PP->L: Largest hash in hash chain for the room
```

*It should be noted that initially, the array of hashed passwords for staff members is empty. When a new staff member joins, they will get their new keycard with their password, and then have to go to each lock individually and use the PP to write the hash of their password into the index of the array corresponding to their id number.

Keycard Configuration

Each keycard recieves the following information from the encoder:

```
1: An is_staff bit
2: The guests checkout time
3: The largest hash in the chain for the room stored on the encoder
```

The size of the information stored on the keycard is 1 bit + 64-bit UTC timestamp and 256 hash for a total of 321 bits, thus under the 416 bit requirement.

Here is how the encoding works in the following situations:

(a) Creating keycards for guests when they first check in:

When a keycard is made for a guest upon checkin, the hotel staff member sets the is_staff bit to be false, tells the encoder the room number the guest will be staying in and the guest's checkout time. The encoder will then write the is_staff bit to the keycard, the checkout time, and the largest hash for that room number. After writing to the card, the staff member will then remove the largest hash from the chain stored in the encoder.

(b) Creating duplicate keycards at that point or later (e.g., if they want a keycard for a roommate):

When a duplicate keycard is made, the encoder is just used to copy the information to a new keycard. This allows both cards to be valid for use.

(c) Creating keycards when the guest believes theirs has been lost or stolen:

In this case, the hotel staff member follows the exact paradigm it does when it first created the keycard for the guest. Remember, since the largest hash on the encoder was decremented after the lost or stolen card was made, the hash on the new card is different. Thus ensuring that the lost or stolen keycard becomes revoked, such that it no longer will open the door. More on this in the design goal 1 section below. It should be noted that in the case of a lost keycard, all current keycard will be invalidated, i.e. roomates or spare keycards will need to be re-issued as well.

Staff Keycard Configuration

Each keycard recieves the following information from the encoder:

- An is_staff bit
- A staff identifier
- The staff members password

The size of the information stored on the keycard is 1 bit + 7 bit staff identifier and a password that is a maximum of 256 bits for a total of 264 bits, thus under the 416 bit requirement.

Here is how the encoding works in the following situations:

(a) Creating keycards for staff when they are first hired:

When a keycard is made for a staff member, the hotel staff member sets the is_staff bit to be true, tells the encoder the said staff member's identifier, and the staff member's password. Then, the encoder writes the information to the keycards magstripe.

(b) Creating keycards for staff when they are lost/stolen:

When a replacement keycard needs to be issued, the exact method used to create a keycard for the staff member when they were hired is used except the staff member/the encoder will create a new password for them. In addition, the staff member will have to go to each lock individually and use the PP to write the hash of their password into the index of the array corresponding to their id number.

(c) Invalidating keycards for staff when they are fired:

Since we cannot assume that the keycard is physically present and thus taken away when a staff member is fired, we follow a similar protocol to when a card is lost. A new keycard is issued using that staff member's identifier and a new password. Then a still-employed member of the staff will have to go to each lock to update the stored hash for the staff member.

Authentication Protocol

Abbreviations:
K = Keycard (to the lock, both staff and guest keycards are the same) , L = Lock

```
1: K -> L: They keycard sends all of its information to the lock

2: L: Check if staff member
   If not staff member:
        -Check that the guest check out time has not passed
        -Check if the hash(hash(hash on the keycard)) OR hash(hash on the keycard) or
   the hash on the keycard is equal to the stored hash
              -If true, update the stored hash to be the same as the hash on the card
   Else:
        -Make a hash of the staff password from the keycard
        -Check that the hash made is equal to the hash stored at the index of the array
   of hashed staff passwords stored on the lock corresponding to the staff members id

3: L: If all checks passed
        -Grant access to the room
      Else:
        -Deny access to the room
```

Design Goal 1

Design goal 1 has been satisfied using the hash chaining system with the keycards as well as the checkout times. After a new guest or a guest replacing their keycard has used their

keycards successfully, the lock updates the stored hash for the room and no longer accepts older versions. In addition, if the new guests never use their card, a guest that should no longer be able to access the room still does not have access because their experation date would have passed. To this point, by checking up to two hashes of the hash on the keycard, we fix synchranization problems that may arise if a guest never uses their keycard. Beyond one guest not using their keycard in a row, more mechanisms are needed. In addition, when the new guests use their card after the skipped guest, all old cards, used or unused, will no longer become accepted due to the storing of the newest hash.

Design Goal 2

When a keycard is inserted into a lock the first thing it checks is whether it is a staff members keycard or a guests. If its a guests, 3 SHA-256 hashes will be performed for a total of 6 ms. If it is a staff member, a single SHA-256 hash is performed for a total of 2 ms. All other operations are checks/comparisons and time is negligible. Therefore, we have well satisfied the required response time.

Design Goal 3

For this design goal, the system successfully prevents guests from accessing other guests' rooms via the unique hash chains used for each individual room. In addition, guests may not deprive other guests of access to their own room because if they insert their keycard into any room that is not their own, it will be denied and no information on the lock will be changed. Lastly, our system follows that it should be highly unlikely that a guest's keycard would be able to open other rooms at any hotel using the system due SHA-256 being highly collission resistant.

As far as technically capable guests, if keycards are stolen, they have the time before a new keycard is swiped with a lock to access the room. Same is true when staff keycards are stolen, they have the time to use the card before the staff member updates all the locks with their new password hash. The other issue is that they have a huge amount of control of the way that program can run if they have the ability to use their own portable programmer to overwrite information on the lock. We believe that security cameras would be a reasonable defense/deterrent against such activities that need physical access to the lock.