CS181S – Systems Security Fall 2020

Problem Session 8: Human Authentication

Monday, October 19, 2020

1. **Passwords.** In this week's lecture videos, we discussed the standard solution to storing passwords: hashing and salting. However, this design generally defends against offline guessing attacks, it is certainly not perfect. Consider the two following additional requirements. For each, (1) design an authentication protocol that would satisfy the additional requirement (be sure to clearly define what information the server will need to store); (2) discuss the positives and negatives of the proposed protocol compared to standard hashing and salting; and (3) decide whether you would be in favor of switching to your proposed protocol.

   (a) Password files should not store plaintext usernames.

   (b) Password-based authentication should be typo-resilient. That is, it should accept if the edit distance between the entered password and the correct password is at most 1.

   Hint: You may assume for simplicity that all passwords are 8 characters and that there are 96 typeable characters.

2. **Tokens.** Your task in this problem is to complete the design of an authentication system for a hotel. The hotel uses an electronic lock system to control access to rooms throughout the property. The system uses the ”something you have” paradigm for authentication. We assume you are familiar with the kind of lock system typically employed in modern hotels, and essentially we are asking you to design that kind of system. The hardware constraints and design goals are given below. The specifics of what you needed to do are spelled out below in the section titled ”Questions”.

   Hotel guests are issued keycards that authenticate to locks. A guest’s keycard should open their room, but not other rooms. We thus omit the issue of common access to the pool or side entrances. Staff are issued master keycards, which can open every lock in the hotel. We thus omit the issue of staff having differing privileges.

   You may assume that the hotel has at most 600 rooms spread across at most 10 floors, and that there are at most 100 staff employed in any given six month period, including all hirings and firings during that time. You may also assume that the expected lifetime of the system is 20 years.

   Hardware Constraints: The authentication hardware involved is as follows:

   - Humans are issued *keycards* as authentication tokens.
   - Rooms have electronic *locks* which require insertion of a keycard to authenticate.
   - The front desk has a stationary computer called an *encoder* that is used to configure keycards.
   - The front desk has a handheld computer called a *portable programmer* that is used to configure locks.

   There is no network in the system: the encoder is not connected to a network, nor are any locks.

   Keycards have a magnetic stripe (magstripe) used to store information. The magstripe can be read and written by inexpensive hardware/software products. The storage capacity of the magstripe is 416 bits. Yes, bits. Storage space is quite limited on the keycard.

   Locks execute a program stored in ROM. They additionally have 16 KiB non-volatile RAM that can be used for data storage. Locks are not tamper resistant: an adversary with physical access to a lock can extract and change any information stored in its RAM, as well as read (but not write) its ROM.

   Locks have two interfaces. The first interface is where users insert keycards; guests and staff use it to authenticate to locks then enter rooms. This *keycard interface* is capable of reading magstripes, but not writing them. The second interface is a port to which the portable programmer can be connected using an inexpensive, standard cable. Staff use this *programmer interface* to perform any other communication with a lock, such as configuration during installation, maintenance, and diagnostics. The port is located on same side of the lock as the keycard interface, hence is accessible from outside the room that the lock guards.

   Locks have a clock that can be set with the portable programmer. Twice a year (shortly before the two nights when daylight saving time changes), hotel staff take the portable programmer to each lock to correct its clock and update its knowledge of when daylight saving changes (since neither those dates nor their calculation is firmly fixed). The clocks drift at a slow enough rate that this biannual visit is sufficient to correct for any errors.

   The microprocessor inside a lock is relatively slow. Computing an AES-128 encryption or decryption of a single 128-bit block requires 1 ms. Computing a SHA-256 hash of message m requires 1 +

ceil($|m|/512$) ms, where $|m|$ denotes the length of m in bits. Those are the only cryptographic primitives available for use, but you are free to build more sophisticated algorithms out of them. Of course, simple bit operations like addition, xor, etc. are available and are much faster than the block cipher and hash operations. So the cryptographic primitives are likely to dominate the running time of whatever you design. Locks are not capable of generating cryptographically-strong random numbers.

Locks are powered by batteries that hotel staff periodically replace. The expected lifetime of a fresh set of batteries is two to four years.

The encoder is a general-purpose personal computer running a standard operating system. You are not responsible for designing the authentication system employed by that operating system. The encoder is connected to a peripheral that can read and write magstripes. The encoder executes a custom application for managing the electronic lock system. When a guest requests a keycard, the hotel front desk staff use the encoder to write information to a keycard, which might be reused from a previous guest.

The portable programmer is a custom piece of hardware and software, but it is built out of standard parts. When not in use, it is kept in a safe behind the front desk; you are not responsible for designing authentication to that safe. The portable programmer has a general-purpose CPU, 16 GiB of flash memory, rechargeable batteries, no cryptographically-strong source of randomness, and limited I/O capability. For the user interface, there is a small LCD display and a keypad, similar to an electronic calculator. For interfacing with locks and the encoder, it has a port for connecting a cable. The portable programmer does not have an interface to read or write magstripes. The typical workflow with the portable programmer is to connect it to the encoder, download some information, disconnect, take the programmer to a lock, connect, perform some operation upon the lock that involves reading/writing the lock's memory, and possibly reconnect the programmer to the encoder afterwards.

Design Goals: Your task is to design an authentication system that satisfies the following goals:

1. When a room transitions in occupation from one guest to another, all that should normally be necessary is for the front desk to encode a new keycard, and for the new occupant to use the keycard to enter the room; at that point, the former occupant's keycard should no longer be accepted by the lock. This goal should be satisfied even if the former occupant never entered the room. (Maybe they rented the room but actually spent the night in another room with someone else. Maybe they wouldn't want their significant other to know that.) But it's rare that more than two occupants in a row would fail to ever enter the room, so it's okay in that scenario for additional mechanisms to be needed.
2. A lock should respond within 100 ms to any attempt to authenticate to it by keycard, either by guest or by staff. Together with the contraints on the microprocessor, this implies that the amount of cryptography that can be used is bounded. In other words, the system should be responsive, so that users don't complain about it.
3. Malicious guests are a possibility. The system shall prevent guests from accessing other guests' rooms, and from depriving other guests of access to their own room. It 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. As usual, assume Open Design. Hence, technically capable guests could build their own portable programmers and magstripe readers/writers, or purchase them (perhaps on the gray market). But your solution does not need to defend against attacks that involve physical modification to locks (e.g., skimmers to read magstripes, overloading the lock with electricity

to burn out the circuits, etc.), nor attacks that involve guests getting access to the encoder or portable programmer (e.g., distracting the front desk staff and using it themselves).

4. Lost keycards are an expected phenomenon for guests, and therefore should be easy to handle. Reenabling room access under a new keycard while revoking access under the lost keycard should at most require the front desk staff to issue a new keycard, and for the guest to authenticate to the lock with that new keycard.
5. Lost keycards (or revocation because of firing) is an uncommon phenomenon for staff, and therefore may be more expensive to handle. Recovering from such a loss may require a staff member to visit every lock that the keycard was capable of opening. But it should not require having to replace the keycards issued to the rest of the staff.

Questions: Given the hardware constraints and design goals above, complete the design of the authentication system. To do that, figure out (i) what will be stored where, and (ii) what protocols will be used in the various tasks mentioned above. Then answer the following questions.

In your solution, bear in mind that you are balancing security and usability. The needs of a hotel are not the same as a military installation. Your authentication system needs to do a good job of achieving security while providing the level of usability that hotel guests expect.

(a) Briefly summarize the design of your system.

(b) How will the encoder be configured after it is installed at the hotel? Describe any initialization that is necessary for the authentication system. (You don't need to discuss configuration of the OS, etc.)

(c) How will a lock be configured by the portable programmer after the lock has been installed in a door? Specify the protocol between the encoder, portable programmer, and lock.

(d) How will guest keycards be created? Specify what the encoder does, especially what it writes on the keycard. Your answer should address creating keycards for guests when they first checkin, creating duplicate keycards at that point or later (e.g., if they want a keycard for a roommate), as well as creating keycards when the guest believes theirs has been lost or stolen. In the latter case, your answer should ensure that the lost or stolen keycard becomes revoked, such that it no longer will open the door. You do not need to address how the front desk staff authenticates the human who is the source of these requests.

(e) How will staff master keycards be created? Specify what the encoder does, especially what it writes on the keycard. Your answer should address creating keycards for staff whey they are first hired, as well as creating keycards when the staff member loses theirs or is fired. In the latter case, your answer cannot assume that the staff member or keycard is physically present.

(f) How will keycards authenticate to locks? Specify what the lock does with the information it reads from the keycard. Be clear about when the lock opens, and when it remains locked.

(g) Explain how your solution satisfies Design Goal 1 (revocation of access from former occupants).

(h) Explain how your solution satisfies Design Goal 2 (timely response from the lock).

(i) Explain how your solution satisfies Design Goal 3 (defense against malicious guests).