Security of embedded devices:

In use of embedded devices, we often want an owner to control a device. For example, your smartphone might be the "owner" for your DVD player. Commands from other parties (including previous owners) are to be ignored by the device.

(a) Explain how such a device might be programmed to recognize commands from its owner, and only its owner. Show how this can be done with public-key technology.

Solution: (There can be more than one solution for this problem)
Let the owner be called A and the device C

1. *The device should have Internet access and some trusted authorities (CA's) public signature embedded within.*
2. *When A passes any command, it should send it as follows:*
   A ----> C: R, CMD, Signature(hash(CMD)),A's Certificate
   (CMD: command; R is a monotonically increasing nonce)
3. *C should verify the certificate using the CA's publick key. Given this, the signature of the hash of the command indicates that this message indeed came from A.*
4. *R ensures that this is not a replay. Since it is monotonically increasing, the device C needs to remember only the last nonce seen (not all of them).*

(b) Explain how such a device might be programmed to recognize commands from its owner, and only its owner. Show how this can be done with secret-key technology.

Solution: It is possible to solve this using a trusted third party (KDC) but to keep things simple I am assuming that a well-shielded cable can be used to communicate securely with the device in case of performing certain functions that happen only once in a while.

1. The device has the shared key of the first owner installed using the cable.
2. *When A passes any command, it should send it as follows:*
   A ----> C: R, CMD, MAC(CMD))
   (CMD: command; R is a monotonically increasing nonce)
3. *C should verify the MAC of the command which indicates that this message indeed came from A.*
4. *R ensures that this is not a replay. Since it is monotonically increasing, the device C needs to remember only the last nonce seen (not all of them).*
5. Note the commands do not need to go over the secure cable as such. They can happen in an environment where eavesdropping can happen.

(c) Describe a protocol explaining how "transfer of ownership" can be accomplished, under the public-key framework of your answer in part (a).

Solution: Assume the new owner is B. To transfer ownership of device C from owner A to owner B, owner B has to sends its certificate to A which A should verify . A should then executes the command "CMD
= XFER, B, Publick-key-of-B." This causes the device to replace PK of A with PK of B, effectively revoking the old owner and enabling the new owner. The command is

protected by integrity as shown in (a).

(d) Describe a protocol explaining how "transfer of ownership" can be accomplished, under the
classical (symmetric-key) framework of your answer in part (b).

Solution:
The new owner B sends over a secure link (could be face-to-face or by other means) to the old owner A the message (B, H) where H = hash(KBC) where h is a secure hash function and KBC is the symmetric
key that the device will eventually receive.

The old owner A sends to the device C the command "CMD = XFER1, B, H". This tells the device to only accept the next command over a secure link iff the initiating party knows a pre-image of H. This prevents other parties from becoming the owner. Note, this command does not have to happen over a physically secure link. The new owner B then connects to the device C over a physically secure link (once
again, something like a cable or shielded IR) and sends the command "XFER2, KBC." The device verifies that this key is a pre-image of H. For future commands over physically insecure links, the device only accepts commands MAC'd with this key. This effectively revokes the old owner's access.