UCCS2

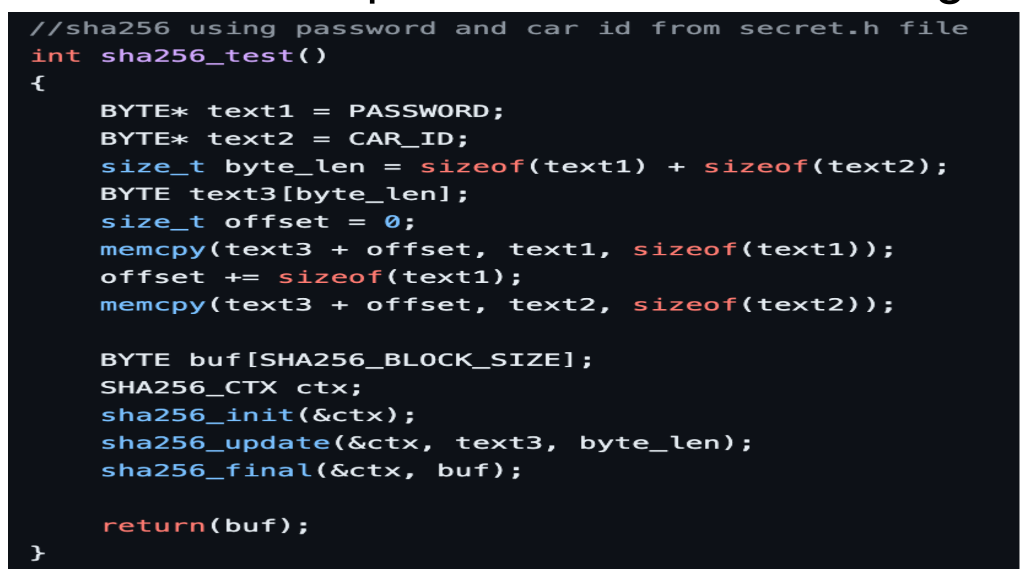
University of Colorado (Colorado Springs)

Ken Lew, Sourav Purification, Arijet Sarker
Advised by: Dr. Sang-Yoon Chang, PhD
April 24, 2023

eCTF

Design Overview

- The protocol is simple hashing of the password and car id that both are computed in the key fob and car. When the key fob initiated the unlockCar() function, it will send the hash to the car for validation.
- SHA256 [1] was used in our design and both firmware would have similar implementation as following:

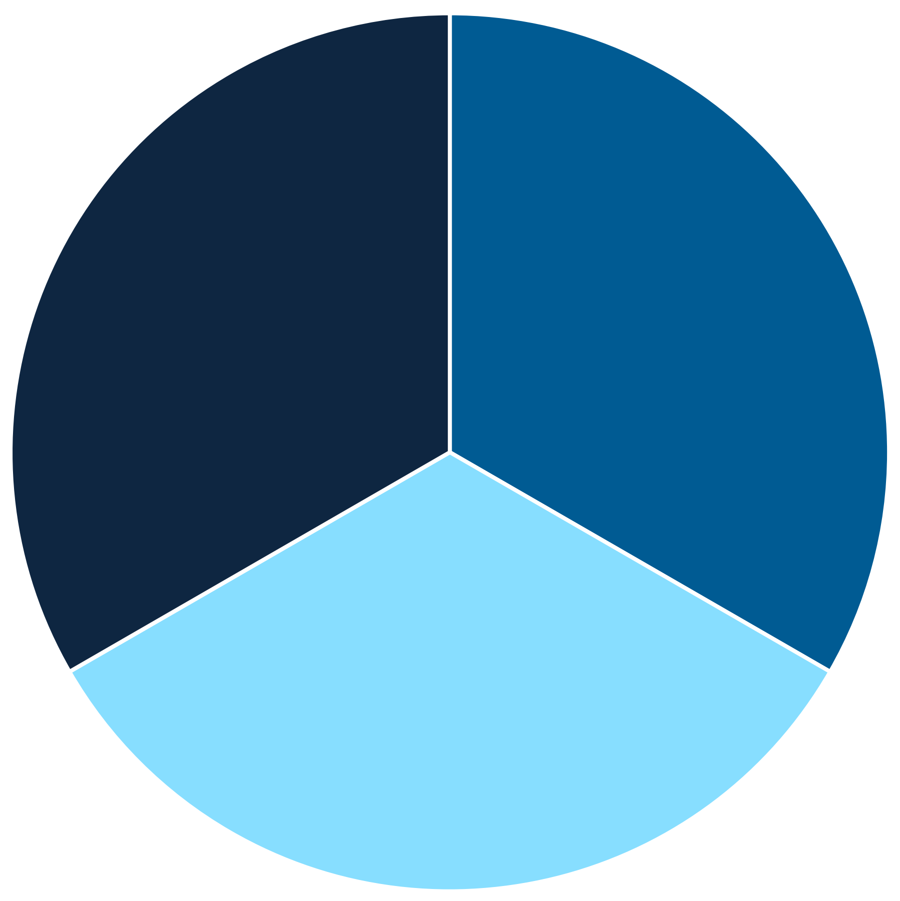


```
//sha256 using password and car id from secret.h file
int sha256_test()
{
    BYTE* text1 = PASSWORD;
    BYTE* text2 = CAR_ID;
    size_t byte_len = sizeof(text1) + sizeof(text2);
    BYTE text3[byte_len];
    size_t offset = 0;
    memcpy(text3 + offset, text1, sizeof(text1));
    offset += sizeof(text1);
    memcpy(text3 + offset, text2, sizeof(text2));

    BYTE buf[SHA256_BLOCK_SIZE];
    SHA256_CTX ctx;
    sha256_init(&ctx);
    sha256_update(&ctx, text3, byte_len);
    sha256_final(&ctx, buf);

    return(buf);
}
```

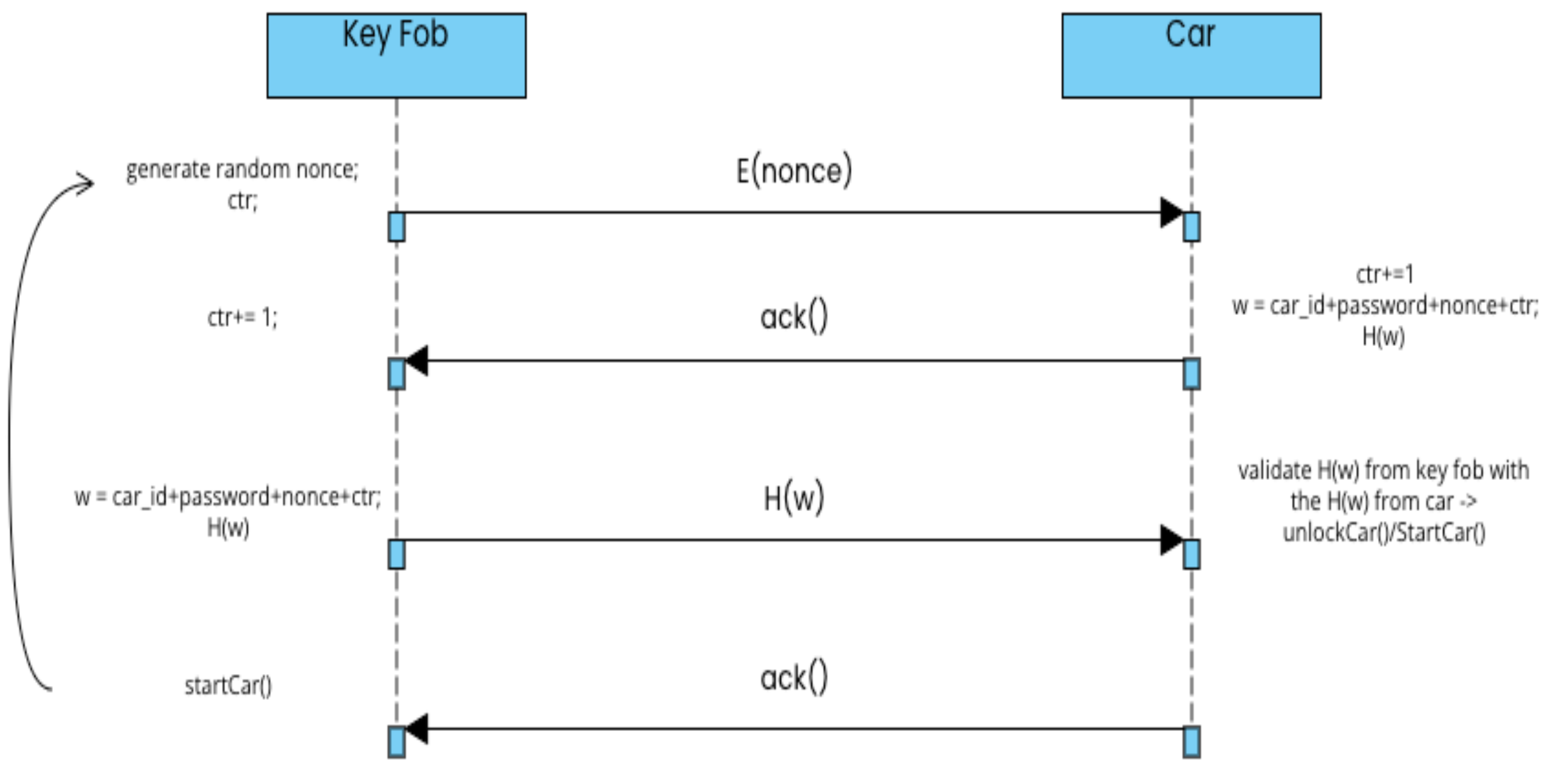
Figure 1: Solved Challenges



■ Leaked Pairing Pin ■ New Car Unlock ■ Passive Unlock

Defensive Highlight

- Since the unlockCar() by default validates only "password", validation measures needed to be put in place like "car id" in order for the car to recognize its own id that a paired fob is only able to unlock the specific car. Hence, "car id" is included in the validation.
- For more measures, hashing and encryption should be included. Due to our own constraint timeline, we were only able to implement hashing. This comes with security vulnerabilities as our car id and password are hard coded.
- If time were not an issue along with technical knowledge during the build phase, we would have implemented this design:


Key Fob
Car
generate random nonce; ctr;
E(nonce)
ctr+= 1;
ack()
ctr+=1 w = car_id+password+nonce+ctr; H(w)
w = car_id+password+nonce+ctr; H(w)
H(w)
validate H(w) from key fob with the H(w) from car -> unlockCar()/StartCar()
startCar()
ack()

In this design we will use encryption like TinyAES [2] to encrypt the nonce sent by key fob. Once car received the nonce, it will send an acknowledgement where key fob will increase the base counter by one. Likewise for car, it will then hash the counter along with car id, password and nonce. When key fob have the same computation and send the hash for car to validate, it will then unlock or start the car depending on the function.

Offensive Highlight

- During the design and built phase, the most fundamental question would be the validation between the cars and key fobs. There were 6 cars total in which each have different car id.
- If so, in reality, a paired key fob should not be able to have access to other cars that are the same model but different cars. If in the car firmware does not validate car id sent by the key fob, then any paired key fob should be able to unlock any cars that have the same firmware configuration (ie: same model or year).
- Therefore, without validation of the car id, security requirements (SR) that impact most in this simple task of loading a paired fob to unlock a car would be Car 1, Car 3 and Car 4. The unlock is completed without much effort.
- The offensive strategy carried out in this competition due were not sophisticated, however very effective. To counter this, all is needed is to implement car id validation to unlockCar() as well as StartCar().
- Although understand that the goal of the competition for SR would be based upon reverse engineering and testing the implementation of cryptography/communication, however, not meeting SR1 would have an avalanche effect on all other SRs.

References

1. https://github.com/B-Con/crypto-algorithms
2. https://github.com/kokke/tiny-AES-c