MITRE eCTF 2023
Final Design
UCCS2

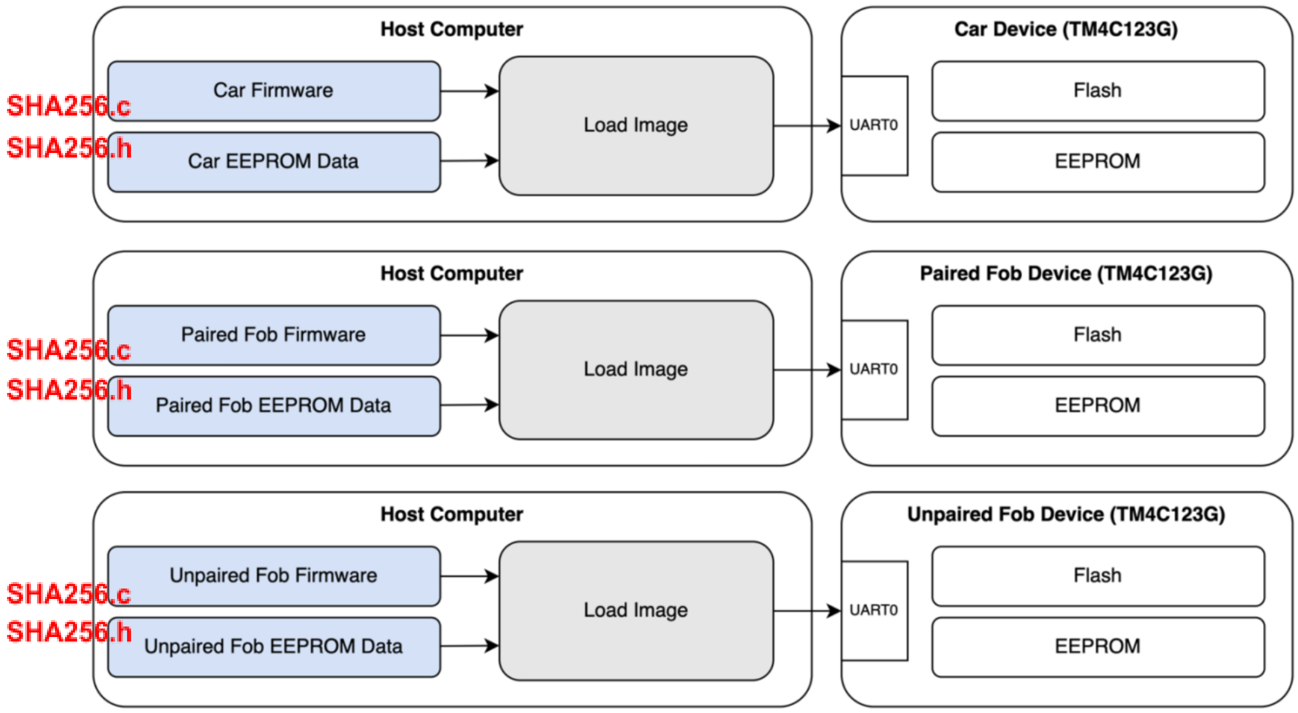
Team members participating in this eCTF would include: Ken Lew (Team Lead), Arijet Shaker, Mark Vaszary, and Sourav Purification

Initial Findings

- The password on gen_secret.py would need to be changed as the parameter intake for unlockCar() function in the car's firmware uses "password" to validate unlocking.
- Only car id is needed for the startCar() function. However, based on documentation, car id is known to the attacker. If so, it is important to have that changed or paired with other data/info known to the only car and key fob where it would provide an additional layer of security.

Documentation

We implemented SHA256 hash in the car and key fob firmware where during the unlockCar() session in the key fob's firmware, a generated hash will be sent over to the car via UART. When the car receives it, in the unlockCar() will do a 'memcmp' of the buffer received from the key fob with the generated hash within the firmware in a function called SHA256_test().



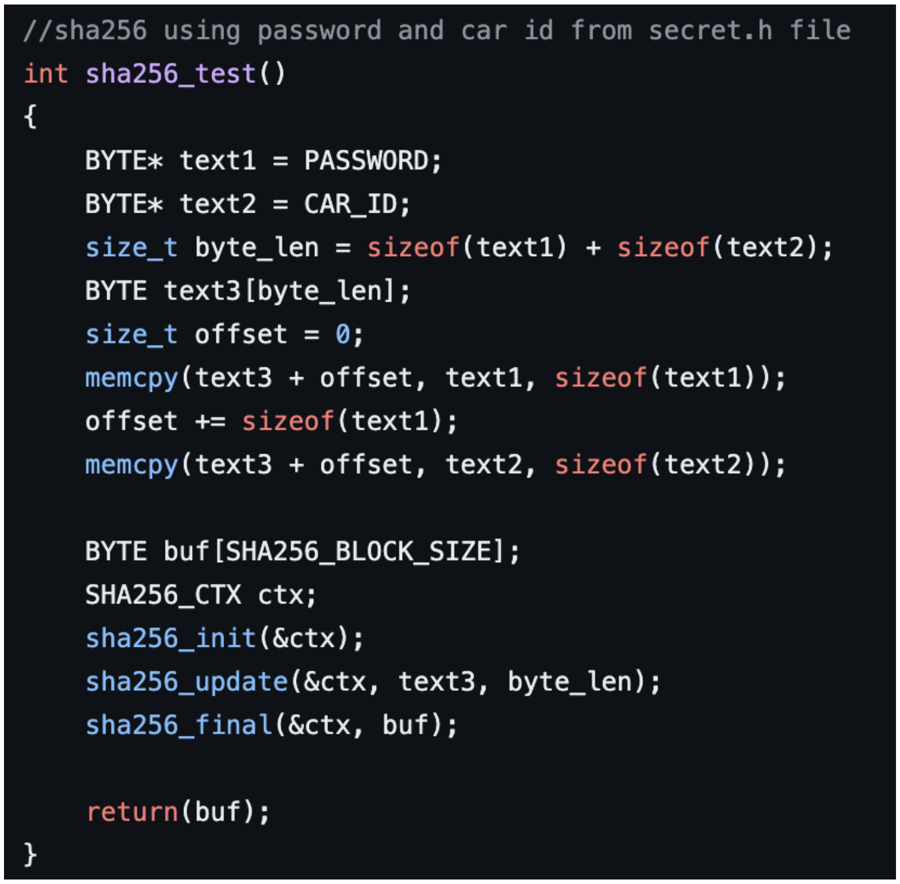
Changes made in the process:

- SHA256.h & SHA256.c - source code SHA256 algorithm (special thanks to Brad Conte and it's code from GitHub)

- #include 'SHA256.h' - In both car and fob firmware
- Car 'firmware.c' - New function name 'SHA256_test()' where this is called during the validation process in unlockCar. The 'SHA256_test()' is being 'memcmp' with the message.buffer in which it is a structure created in car firmware in order to receive the buffer from key fob.
- Fob 'firmware.c' - In the 'unlockCar()' function, I have used text 1 & 2 to store the password and car id that is in a paired fob while concatenating both password and car id into another text called text 3. Using SHA256.h and SHA256.c, I am able to initialize, update and produce a final hash that will consistently update BYTE buf. Using the original 'MESSAGE PACKET', I am able to use SHA256 block size and declare into message length and send the buf via 'message.buffer' variable. On the car side, the car will also be producing the same hash for verification in the car's unlockCar() function.

CODE

In car's firmware, the SHA256 is implemented in a function call:

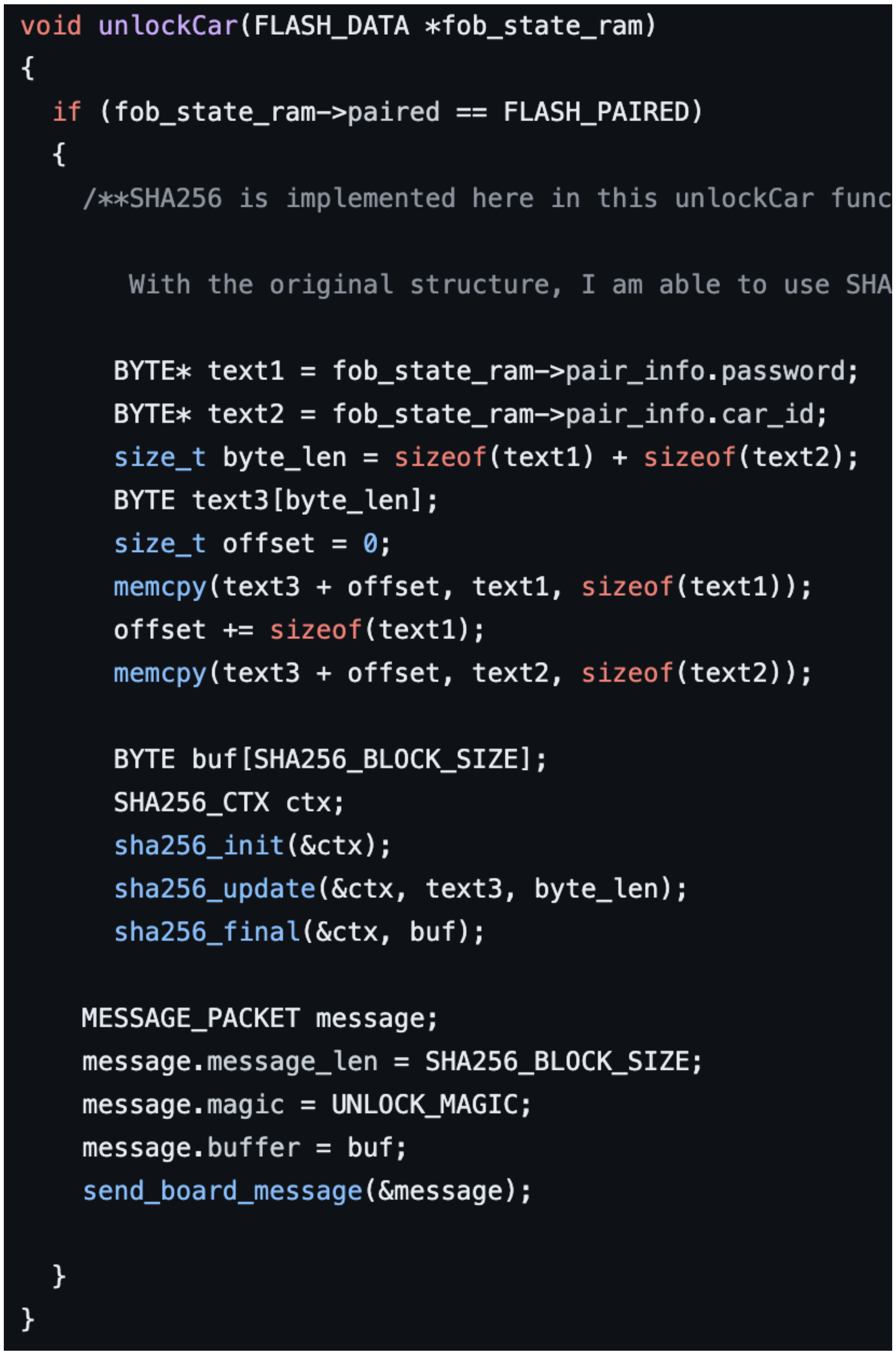


```c
//sha256 using password and car id from secret.h file
int sha256_test()
{
    BYTE* text1 = PASSWORD;
    BYTE* text2 = CAR_ID;
    size_t byte_len = sizeof(text1) + sizeof(text2);
    BYTE text3[byte_len];
    size_t offset = 0;
    memcpy(text3 + offset, text1, sizeof(text1));
    offset += sizeof(text1);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

However, in the key fob's firmware, the SHA256 is implemented within the unlockCar() function. It would be as following:



```
void unlockCar(FLASH_DATA *fob_state_ram)
{
  if (fob_state_ram->paired == FLASH_PAIRED)
  {
    /**SHA256 is implemented here in this unlockCar func

       With the original structure, I am able to use SHA

      BYTE* text1 = fob_state_ram->pair_info.password;
      BYTE* text2 = fob_state_ram->pair_info.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);

    MESSAGE_PACKET message;
    message.message_len = SHA256_BLOCK_SIZE;
    message.magic = UNLOCK_MAGIC;
    message.buffer = buf;
    send_board_message(&message);

  }
}
```

Essentially, I am using the password and car id from the fob state ram to concatenate into a BYTE variable called text3. The text3 variable will be used throughout the sha256 process stored under the variable 'buf' where it will be stored under 'message.buffer' to be sent via UART to car.

I was not able to get to startCar() function due to time constraints on my part,and would expect some vulnerabilities where car id is known to attackers.