Team FAU

Florida Atlantic University

**Rabih Elkhatib, Daniel Owens, Aimee Laclaustra, Mila Anastasova,
Angelina Guasti, Boone Douglass, Dov Cattan, Eadan Plotnizky, Eric Gonzalez, Nelson Menyaga,
Sunny Chen, Wen-Chung Cheng
Advised by: Dr. Rezarderakhsh, PhD, Dr. Rami Elkhatib, PhD**

April 24, 2023

Design Overview

Our design employs AES to create a secure channel through a simplified handshake protocol between a car and its paired fob, as well as when pairing unpaired fobs and creating/enabling features. Data is sent and received with pseudo-randomly generated encryption keys derived from a shared secret between components. In theory, this ensures that the data being sent and received will be randomized every time, making it difficult to intercept and open the car or enable features.

Defensive Highlight

For the design phase, our goal was to fulfill all the security rules we were tasked with[1]. We devised the following measures to accomplish this:

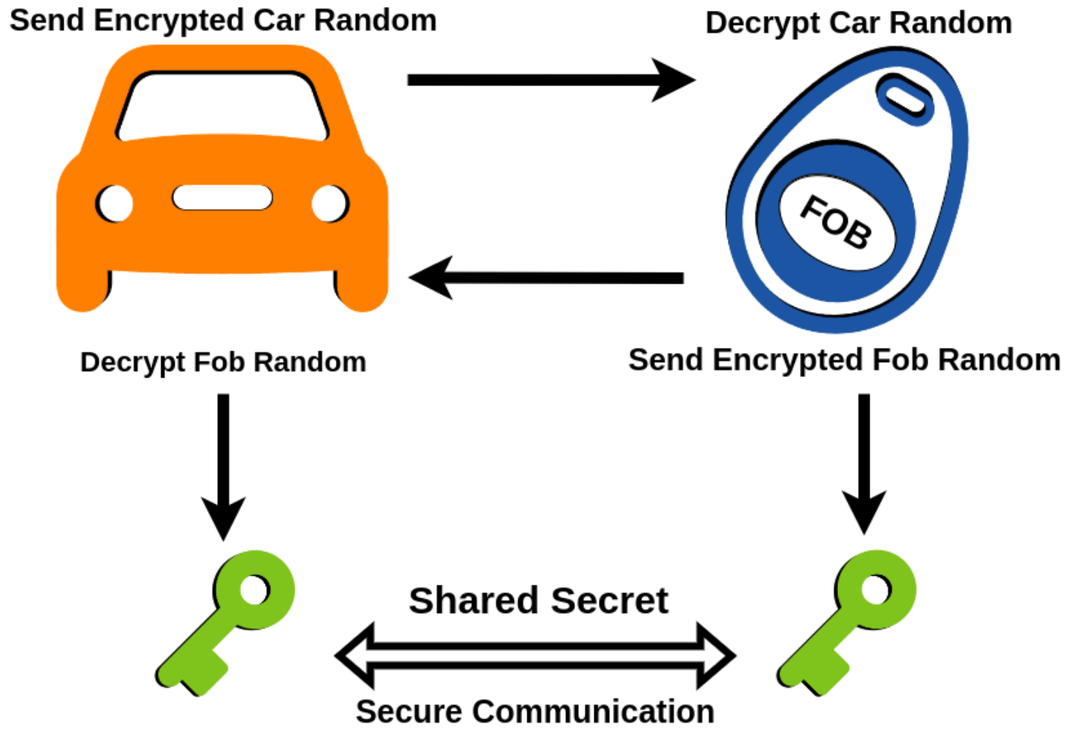
- In order to accomplish SR1 and SR3, we generate 256 unique keys for each possible car ID (0-255) at the deployment phase. Then at the build phase, the key associated with the car's particular ID is then stored in the EEPROM of both the car and its paired fob. To create a secure channel, the car and fob exchange random values encrypted with these keys using AES. If the secret is not the same for both parties, the car and will not unlock or start. Figure 1 illustrates this exchange. The goal is to ensure that each paired fob can only unlock a car and its features that have a specific ID,

- A similar approach is done when pairing unpaired fobs where we use an additional 257 key that is exclusive for pairing and shared between all authentic fobs. This measure is done so that only legitimate unpaired fobs are able to pair properly, and accomplishing SR2 and SR4 in the process

- SR5 and SR6 involve features that are created through a package feature tool, and the only communication required is with a fob. We devised a similar approach using the unique key for each car ID and encrypting the data being sent

One feature we paid attention to was random generation. Initially, we planned on using the board's built-in temperature sensor, and accumulating enough readings to generate a long enough random value. We theorized that attacking this design would be very difficult and would require shorting the sensor. Unfortunately, this idea caused issues with the secure bootloader used during the attack phase, and we were not able to fix this problem in a timely manner. We had to compromise by storing additional random values in EEPROM in the build phase. This is susceptible to replay attacks, so in the future we hope to work on getting the temperature sensors to work and look into EEPROM protection.

Figure 1: Securing Car and Fob



Offensive Highlight

For the attack phase, we mainly targeted designs that did not create a unique link between cars IDs and fobs. This can be exploited by using paired fobs associated with a different car to obtain flags. For example, in some cases fob 0 can be used to unlock cars 1-4 and obtain their flag if no measures are taken against this, despite using a secure channel.

We also attempted replay attacks against designs that had measures against this vulnerability, but the data sent followed a pattern instead of being fully randomized. We used a USB-to-TTL Serial Cable to intercept the data and send to the car ourselves. Unfortunately, we were not able to capture flags with this method.

One attack we would like to highlight is against the enable feature flag where an insecure design will have a very obvious pattern in the feature message. For example, if features 1 and 2 in cars 0-4 follow a certain pattern where only 1 or 2 characters change between the feature messages, then that can be easily used to identify what message for feature 2 of car 5 should look like, allowing the attacker to capture this flag.

We propose a fix to prevent this: In the package tools, the message can be encrypted (using AES or other schemes) with a key stored in a secret directory that will only the fob will also have access to in order to decrypt the message. A random number can be appended to the end of the message before encryption to increase entropy and remove the chance of any patterns being visible. This way the message will be impossible to decipher without knowing the key, and the attacker will need to find another method to obtain the feature flag.

References

1. ectf.mitre.org/wp-content/uploads/2023/02/2023-eCTF-Rules-v1.1-lr.pdf