Key Fob Enthusiasts

U.S. Air Force Academy

C2C Jabari Bowen, C1C Patrick Carattini, C1C Kevin Carrig, C2C Chanon Mallonoo, C2C Payton Rawson
Advised by: Dr. Stanley Baek, PhD
April 24, 2023

Design Overview

Our design called for using symmetric keys stored in host secrets to facilitate encryption for car, fob, and host communication. We chose AES encryption because the library was already provided in the example repo which streamlined implementation. A big challenge for us came in implementing encryption on host-fob communications, since the fob was build in C and the host was built in python. The heterogeneous development environment raised issues since the way the C library encrypted a message may not be the exact way that the python library expected for decryption. Our end solution was instead of using two different libraries (tiny-aes-C and pyaes), we used tiny-aes-c and its cython wrapper for compatibility.

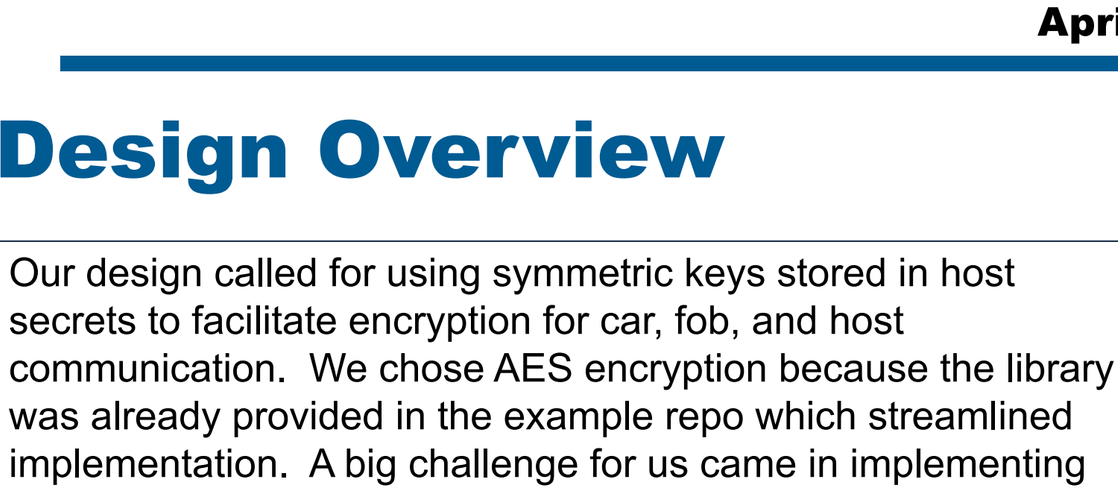


Figure 1: Pairing an unpaired fob.

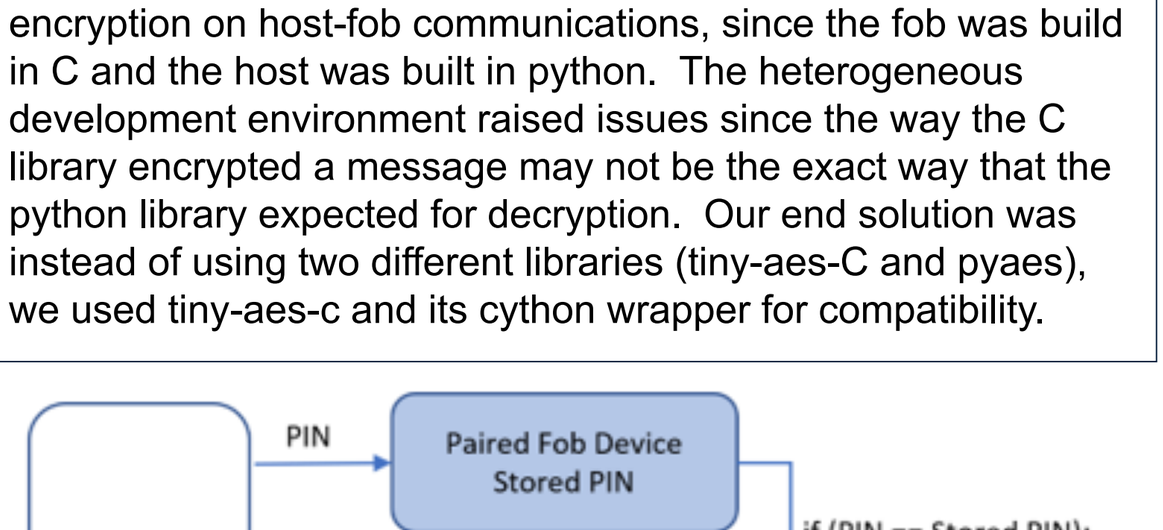


Figure 2: Initial Process to Enable Feature on a Paired FOB

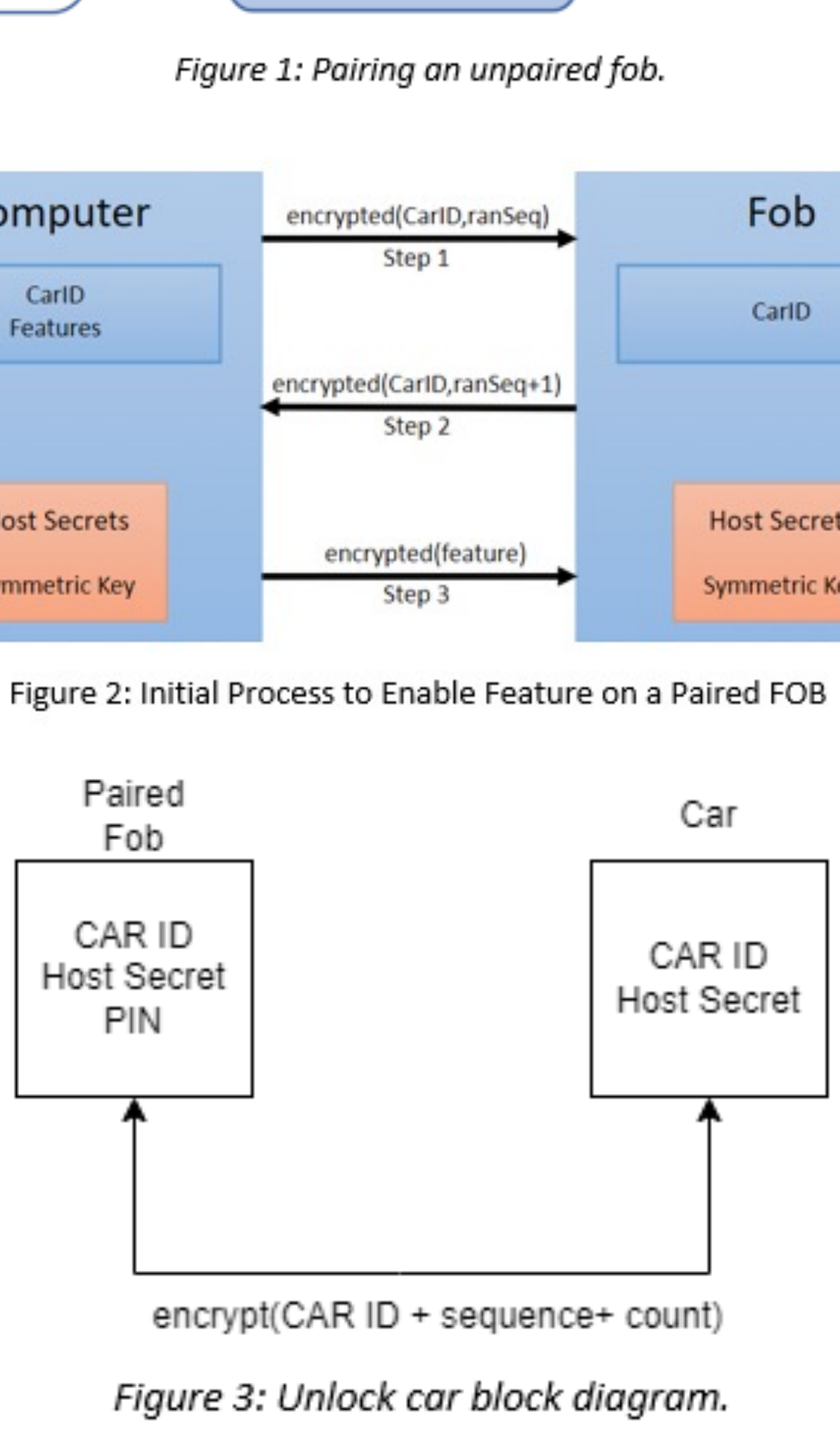
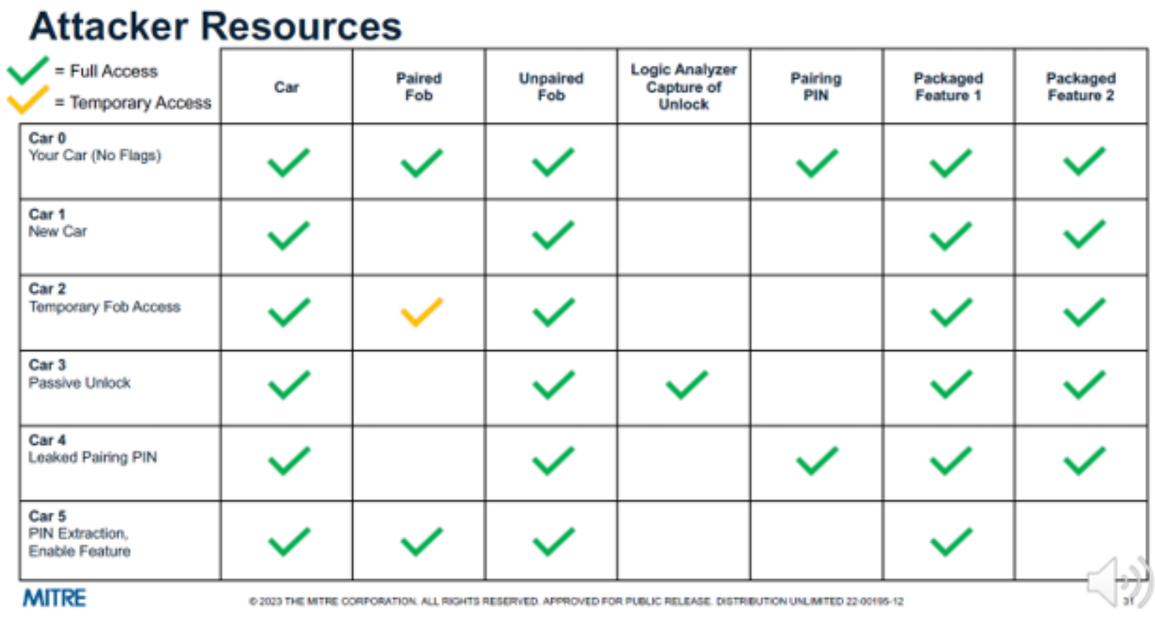


Figure 3: Unlock car block diagram.

Attacker Resources



= Full Access
= Temporary Access

	Car	Paired Fob	Unpaired Fob	Logic Analyzer Capture of Unlock	Pairing PIN	Packaged Feature 1	Packaged Feature 2
Car 0 Your Car (No Flags)	✓	✓	✓		✓	✓	✓
Car 1 New Car	✓		✓			✓	✓
Car 2 Temporary Fob Access	✓	✓	✓			✓	✓
Car 3 Passive Unlock	✓		✓	✓		✓	✓
Car 4 Leaked Pairing PIN	✓		✓		✓	✓	✓
Car 5 PIN Extraction, Enable Feature	✓	✓	✓			✓	

MITRE

Figure 4: Attack Scenarios

Defensive Highlight

Due to hurdles with the development environment, we never submitted a design for other teams to attack. Given the time we would implement a rolling key system, similar to what real car fob pairs use, to make our system resistant to replay attacks.

To account for multiple key fobs and cars we would like to substitute the symmetric key encryption for a pub and private key infrastructure, or a hybrid of the two. Each element (car, fob, host) would have its own randomly generated private and public key pairs on boot using systick, allowing the system to track which car & fob are communicating and have burned codes from the rolling code.

Offensive Highlight

Although we never made it to the attack phase, we discussed different methods we would have used to attack other systems. A man in the middle attack to intercept transmissions from the fob to the car would have been the place to start. Some teams discussed physical attacks including de-soldering the eprom and 'reading it manually. While potentially effective, the limited amount of boards available to us limited our thinking to less invasive methods. Replay attacks are very effective against real car-fob pairs, given that the signal is captured when the fob is out of range from the target fob, so the car doesn't know that the code has been burned from the rolling code. A similar method could be attempted in this competition where we capture a UART signal without the target board connected, then replay it to gain access.

References

Fatima, S., Rehman, T., Fatima, M., Khan, S., & Ali, M. A. (2022). Comparative analysis of AES and RSA algorithms for Data Security in cloud computing. *IEEC 2022*. https://doi.org/10.3390/engproc2022020014