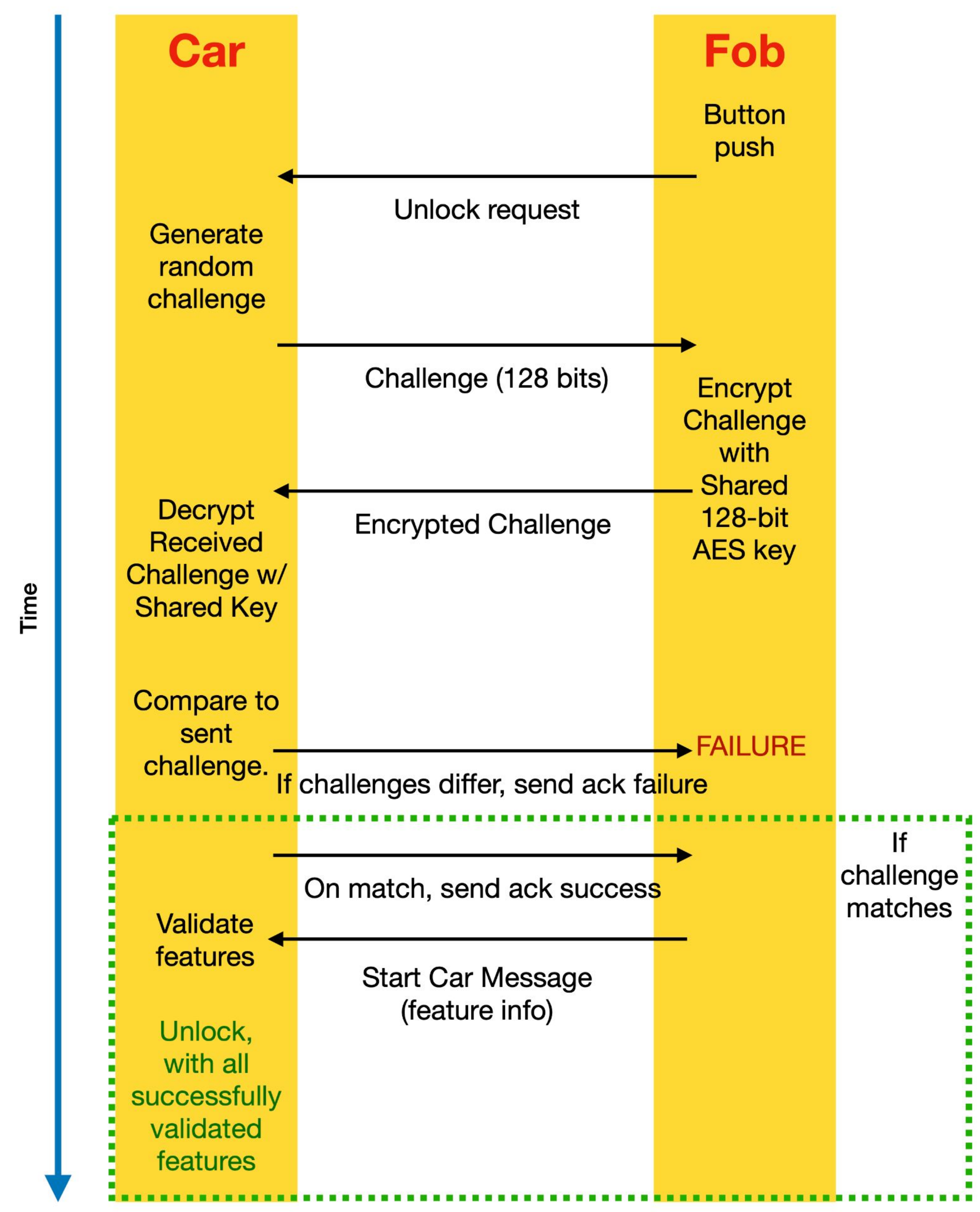
Tufts

Tufts University

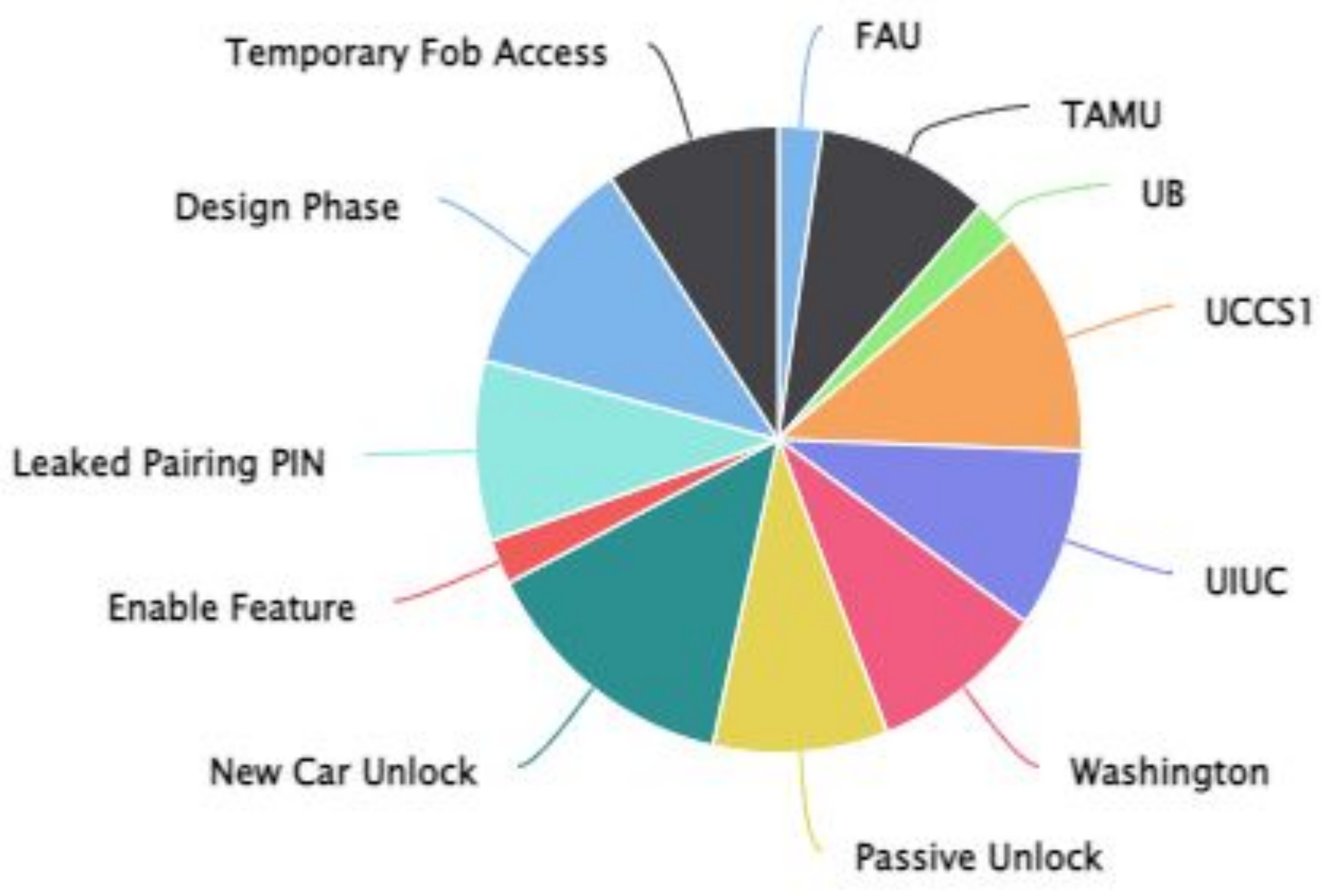
Elliot Bonner, Felipe Camargo, Adrien Lynch, Harshdeep Komal, Rusny Rahman
Advised by: Ming Chow
April 24, 2023

Design Overview


Car
Fob
Time
Button push
Unlock request
Generate random challenge
Challenge (128 bits)
Encrypt Challenge with Shared 128-bit AES key
Encrypted Challenge
Decrypt Received Challenge w/ Shared Key
Compare to sent challenge.
FAILURE
If challenges differ, send ack failure
If challenge matches
On match, send ack success
Validate features
Start Car Message (feature info)
Unlock, with all successfully validated features

Our design uses an AES-based challenge response system; the challenge is randomized based on timers and seed data in EEPROM (replaced every power cycle).

Features are protected by computing a SHA256 hash of the feature number, then encrypting using a car-specific key. The car decrypts the feature package and can use the hash to confirm validity.



Defensive Highlight

1. *Security measures in our design*
   - Tiny-AES library to encrypt & decrypt random challenge exchanged between car & fob, to confirm shared key valid before unlock
   - Incorporating random number generation via microcontroller timers (unique seed) as well as seed data in EEPROM
     - To seed the RNG, the existing EEPROM seed plus timer data is used, then a new set of random data is written to EEPROM
   - Verifying message length before loading into buffer (prevent buffer overflow)
   - Encrypting a feature packaging message as one AES block containing the feature in the last 8 bits of the message and the leftmost bits of the SHA256 hash of the feature number in the remaining bits of the message.
   - Using constant-time checks when comparing data
2. *Highlight one defensive feature, explaining why you decided to include it and what it is (or was) supposed to accomplish*
   - The feature packaging format encrypts features using a car-specific key, validated by the car rather than the fob. This ensures the feature is only valid on one car; on any other, the decrypted feature will be garbage. By including a hash of the feature, the car can verify with a high degree of certainty that the feature was not encrypted using a different key (this acts as a MAC).

Offensive Highlight

1. *A brief summary of some of the attacks that your team developed*
   - Taking advantage of recurring use of keys generated via global secrets
   - Feeding all possible pairing PINs into the paired fob from a computer
   - Looking at copies of global secrets in the fob0 unencrypted binary (particularly if global_secrets.txt was stored directly in EEPROM as in the reference design)
     - Using such extracted secrets to build a deployment that shared global secrets with the attack deployment and then using this deployment to build and load new features
2. *Highlight one attack and explain what security vulnerability the attack exploited and why the attack was (or wasn't) successful. You should explain your work in enough detail for the reader to reproduce the attack (code snippets are allowed).*
   - In several designs, the keys used were global rather than per-car. This meant that the fob0 image provided could be used to unlock cars 1-4 with no further changes.
3. *A proposed fix to the code that would prevent your attack from working. The fix should be described in enough detail for the reader to implement themselves.*
   - Generating different keys, ensuring global secrets are distinct from car secrets, etc

References

1. http://burtleburtle.net/bob/rand/isaacafa.html
2. TI Microcontroller Datasheet
3. TI Documentation
4. tiny-AES-C
5. https://github.com/B-Con/crypto-algorithms