Design Document - IITM

Security Requirements

SR1: A car should only unlock and start when the user has an authentic fob that is paired with the car

SR2: Revoking an attacker’s physical access to a fob should also revoke their ability to unlock the associated car

SR3: Observing the communications between a fob and a car while unlocking should not allow an attacker to unlock the car in the future

SR4: Having an unpaired fob should not allow an attacker to unlock a car without a corresponding paired fob and pairing PIN

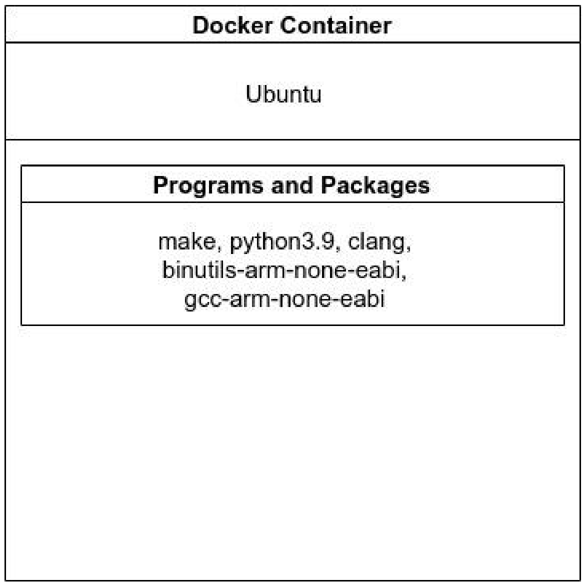
SR5: A car owner should not be able to add new features to a fob that did not get packaged by the manufacturer

SR6: Access to a feature packaged for one car should not allow an attacker to enable the same feature on another car

Design

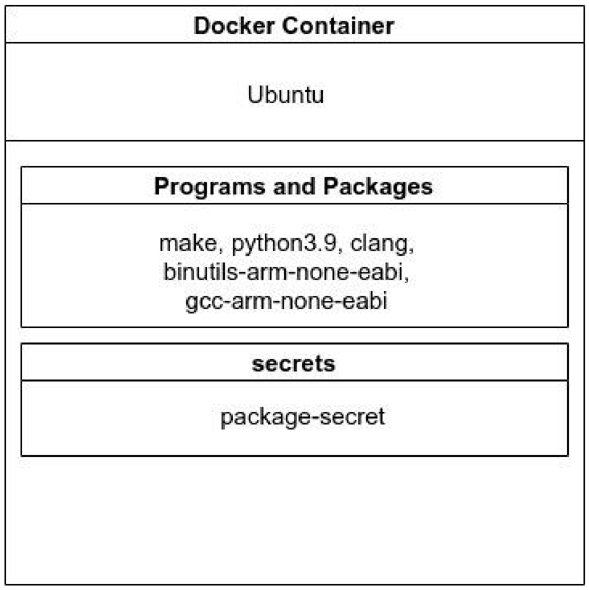
Docker_Env

A docker image as described in the diagram will be generated.



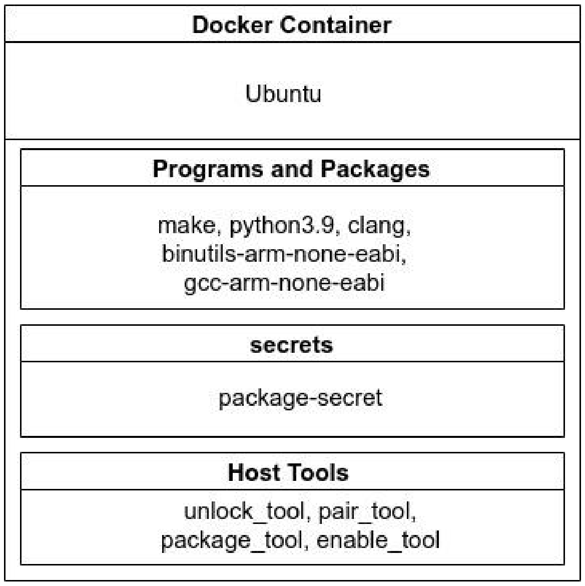
Deployment

generates system-wide secrets used in encryption for various purposes.

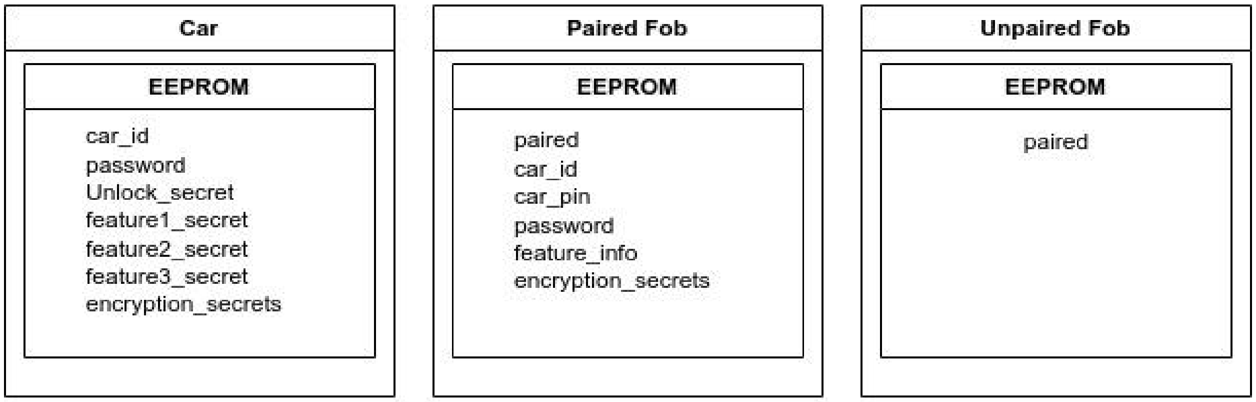


Host Tools

Runs in the docker environment and communicates with the devices using sockets. These tools mainly provide unlock, pair, package and enable functionality.


Docker Container
Ubuntu
Programs and Packages
make, python3.9, clang, binutils-arm-none-eabi, gcc-arm-none-eabi
secrets
package-secret
Host Tools
unlock_tool, pair_tool, package_tool, enable_tool

The following diagram gives the details about each device which will be helpful in understanding the working of each protocol. The encryption_secrets are the keys used for communication between car-paired fob and fob-host machine.



Structure of the Boards Running the Firmwares

Unlock tool

Ideal functionality: The car should only be unlocked by the corresponding paired fob.

Host machine prints unlock messages and feature secrets or failure messages received from the car depending upon whether the car is unlocked successfully or not.

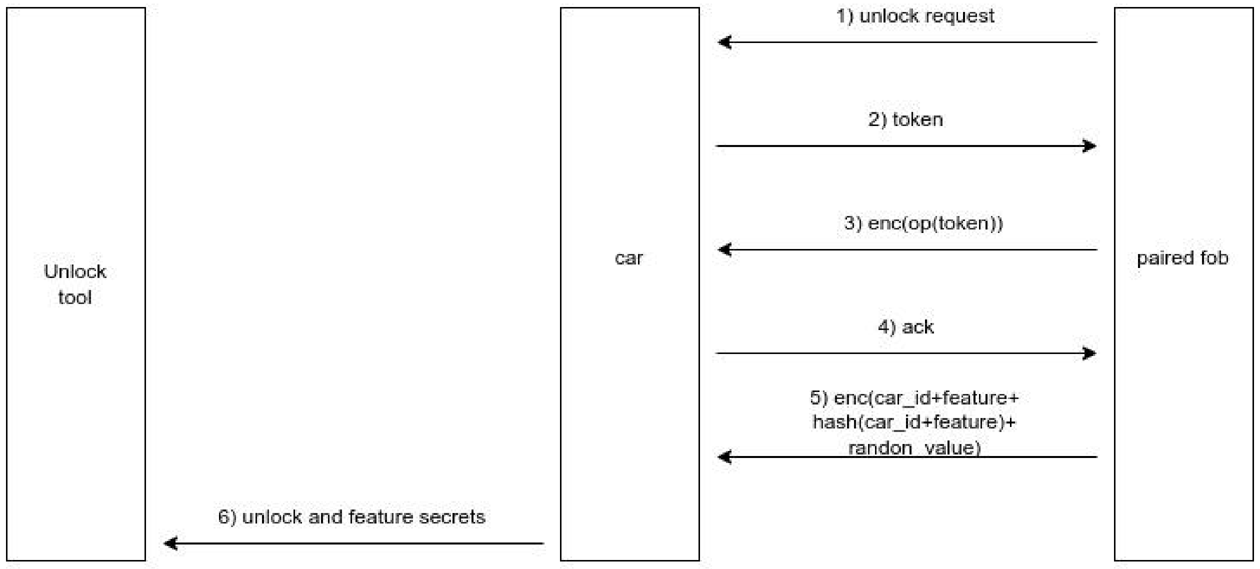
Protocol:

When paired fob sends the unlock request to the car, the car sends a randomly generated token to the fob. Since the Tiva microcontrollers does not have hardware RNG, the seed for the random token is generated using any of the in-built on-chip sensors (Ex: in-built temperature sensor in the ADC).

The fob then performs certain obfuscation operations on the token and sends the result after encrypting it using the key shared with the car during the packaging of the firmware. The car after decryption using the pre-shared key and de-obfuscation, compares the token. If the received token is found to be the same and received under the threshold time limit, then the car is unlocked.

The usage of encryption prevents SR1 while the usage of random token ensures that replay attacks are avoided - ensuring SR2, SR3 are satisfied. While rolling the tokens prevents replay attacks, it is susceptible to roll-jam attacks. To prevent roll-jam attacks, the validity of token is limited to a time-threshold, after which the token becomes invalid.

Once unlocked, an acknowledge message is sent to the fob, which then sends the feature information along with the car id after encrypting. The following figure describes the working of the protocol,


Unlock tool
car
paired fob
1) unlock request
2) token
3) enc(op(token))
4) ack
5) enc(car_id+feature+ hash(car_id+feature)+ randon_value)
6) unlock and feature secrets

unlocking car

Pair tool

Ideal functionality: without the correct car pin and paired fob, one should not be able to pair any unpaired fob to the car.

Host machine sends a pairing signal to both paired and unpaired fobs. It also sends the pairing PIN to the paired fob. Paired fob after verifying the pairing PIN sends the password, encryption_secrets, and feature info along with the car_id and pin to the unpaired fob.

Without a paired fob, and pairing PIN, the unpaired fob can’t be paired with the car. Hence the SR4 is met.

Package tool

Ideal functionality: One should only be able to enable the feature packaged for their specific car and should not be able to activate any other feature or use the package for any other car.

The packaged binary file generated by host machine using car_id and feature_id will contain Enc(car_id+feature_id+Hash(car_id+feature_id)+random_value), where hash and enc are hashing and encryption functions respectively. This ensures that the confidentiality, authenticity and integrity of the contents (car_ID + feature_ID) of the packaged feature

Enable tool

Host machine sends the data stored in the packaged binary file to the paired fob. The fob after decrypting the message compares the hash value of the car_id and feature_id to the hash received from the message. If both of them are same then the fob compares the car_id with the id stored on the fob to enable the feature. This ensures that the feature is authentic and not tampered with in-between.

The encryption secrets used for packaging the features will be generated after deployment and will be shared with the paired fobs. The attackers will only have access to the encrypted binary packages and not the host secrets used to encrypt them. Also, the communication where paired fob sends active features to the car is also encrypted. Hence, SR5 and SR6 will be met as the attacker won’t be able to package and enable the feature without the encryption keys.

Since the TivaWare library provides support for AES encryption, AES will be used for encryptions - provided it is resistant to side-channel attacks.