PARED Design

Provisioning

During the build step, EEPROM images with cryptographic secrets, essential state variables, and hashes are generated for each device. On accesses to EEPROM memory, the corresponding hash of the accessed region is recomputed to ensure integrity. Flash memory is unused and is marked as read-only on startup.

Cryptographic Algorithms

These are the cryptographic algorithms used in our design and their purposes:

- XChaCha20 (symmetric stream cipher with extended nonce)
- p256 (ECC-based signature algorithm – NIST recommended)
- Blake2s (cryptographically secure hash – SHA-3 finalists)

Secrets and Keys

- Text hash: Blake2s
    - On startup, the firmware hashes itself to provide an integrity check for tampering at-rest
- Seed: ChaCha20
    - On startup, the firmware loads and increments a seed from EEPROM (initialized with entropy from the build environment) to provide a CSPRNG for nonce generation
- Pairing key: p256
    - Authenticates the car to the fob during unlock
    - Unique for each car
    - Car has the public key component
    - Paired fob has the private key component
- Package signing key: p256
    - Ensures integrity and non-repudiation of feature packages
    - Private key is only accessible during provisioning
    - Fob has public key
- Manufacturer symmetric key: XChaCha20
    - Establishes a secure channel for transferring secrets during pairing
    - Both paired and unpaired fobs have this key
    - AEAD cipher ensures integrity of message
- Manufacturer signing key: p256
    - Ensures authenticity of unpaired fob during pairing process
    - Private and public key are present on both unpaired and paired fobs

Pairing

The paired fob verifies the PIN supplied by host-tools CLI by comparing the computed hash with the one stored securely in EEPROM, and finally it transfers the necessary secrets to the unpaired fob over a secure encrypted channel. On any failure, the paired fob will assume a brute-force attempt and set a 5 second timeout.

Pairing Process

1. Host-tools sends a pairing command to both fobs.
    a. Host-tools also sends a second command to specify whether the fob is the paired or unpaired side
2. Host-tools sends PIN to the paired fob.
3. Paired fob sets the timeout flag, this provides a "critical section" whereby any failures would be detected between power cycles, ensuring the timeout executes in full.
4. Paired fob verifies the PIN is correct by computing its hash. If verification fails, a 5 second timeout will occur.
5. Paired fob sends secrets to unpaired fob, encrypted by the manufacturer symmetric key.
    a. Secrets are PIN hash, car ID, and pairing private key, and car auth key

Unlocking

The paired fob sends an unlock request to the car, and authenticates itself to the car using a challenge-response system that verifies it has the requisite pairing key. If successful, the car sends flags and package messages to host tools.

Unlocking Process

1. Paired fob sends an unlock command to car
2. Car sends challenge nonce to paired fob
3. Paired fob computes and sends response to car by signing the challenge and enabled features with the pairing private key
4. Car verifies response using the pairing public key
5. Car sends unlock message and messages for enabled features to host tools

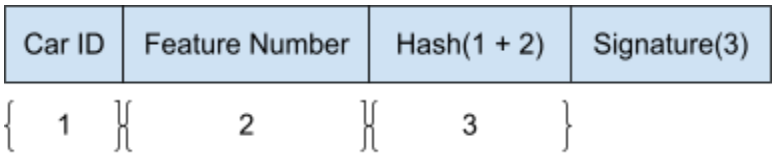
Feature Packages

The “Package Feature” host tool hashes and signs the package (which consists of the car ID and feature number) using a package signing key. Verification of the package occurs on the fob during the “Enable Feature” process.

Package Processes

Package Feature

A packaged feature has the following structure:



Package tool:

1. Read package signing key from secrets
2. Sign and hash the car ID and feature number
3. Write the packaged feature to file

Enable Feature

1. Host tools sends enable feature command to fob
2. Host tools sends feature to fob
3. Fob verifies package signature, hash, and car ID.
4. Fob calculates the new enabled features state, hashes it, and stores the new state and hash in EEPROM