eCTF 2023 UIUC Design Document

Version 1.3

Components

Overview

The security of our PARED design relies on ECDSA, using signatures to validate communication between car and fob as well as signing features. As such, the generation of key pairs are necessary for each component:

- A manufacturer keypair
- A car keypair
- A fob keypair

We use the P256 elliptic curve cryptography in our implementation of the design.

Pairing Fobs

- During the build process, each fob (regardless of paired or unpaired) will receive a unique salt to be stored in EEPROM. This salt is then combined with the correct PIN to form the PIN hash, which is also stored in EEPROM.
- When pairing, the paired fob checks if a provided PIN attempt was correct by hashing the combined salt and PIN attempt and comparing it with the stored PIN hash. If the PIN is correct, then the paired fob begins sending its data to the unpaired fob, including the fob secret, car public key, car ID, and feature signatures.
- To protect against further potential attacks, the paired fob also does not send the fob secret directly to the unpaired fob. Instead, we have a "fob secret encrypted" value, which is a copy of the fob secret XOR encrypted with the SHA256 of the combined PIN and fob salt (opposite concatenation order to prevent reuse of the stored PIN hash). This means that only the correct PIN can properly decrypt the "fob secret encrypted" to be sent to the unpaired fob.
- This pairing protocol finishes within 1 second on a successful unlock, and has arbitrary delays added to ensure that each transaction takes closer to 1 second to prevent brute force attacks. Additionally, if there is an invalid PIN attempt, the paired fob enteres a blocking state for 5 seconds to prevent additional pairing attempts.

Packaging, Enabling, and Verifying Features

- The feature number is concatenated with a unique car ID and is signed at the manufacturer's deployment using the manufacturer's private key.
- Signed packages can then be transported to a paired fob through the enable feature transaction. The fob will not make any attempt to validate whether the package was signed correctly by the manufacturer - this is the car's responsibility.
- After the car is unlocked, the fob will send only the feature signatures in order of feature number 1, feature number 2, and feature number 3. We do not provide the original message these signatures were signed with since the car will use its own car ID and corresponding feature number to verify the signature.
- For each of features 1, 2, and 3, the car will verify the signature using the manufacturer public key stored in EEPROM.

Unlocking and Starting Car

- The car and fob keypair are crucial during unlocking. The car stores its private key and the fob's public key in EEPROM, while the fob stores its private key and the car's public key in EEPROM. Multiple paired fobs will share the same private key to unlock the car.
- The design of unlocking is similar to a challenge-response authentication system. The car randomly generates a nonce and signs it with its private key. The car then sends the nonce value and signature to the fob. Note that this nonce value MUST be generated randomly and in a non-deterministic fashion to prevent replay attacks.
- Upon receiving the nonce and signature, the fob validates the nonce and signature using the car's public key. This is done to prevent oracle attacks.
- After validating the signature, the fob will add 1 to the nonce value and sign that value with its private key. The only reason we add 1 is to prevent the astronomically unlikely possibility that the car and fob keypair are the same. The fob then sends the nonce +1 and signature value.
- The car then validates nonce + 1 and the signature with the fob's public key. Note that the car must not use the message value that was sent by the fob and should compute nonce + 1 itself and use the fob's provided signature. This prevents an attacker from simply replaying any fob signed message to unlock the car.
- If the fob is validated, then the car unlocks and requests features from the fob to finish starting (outlined above in "Packaging, Enabling, and Verifying Signatures").

- This unlock protocol finishes within 1 second on a successful unlock. However, if there is an invalid unlock, the car enters a blocking state for 5 seconds to prevent oracle attacks. Additionally, the unlock process should not happen as fast as possible, with arbitrary delay tofurther prevent brute force attempts.

Randomly Generating Numbers

Given that the TM4C123GXL board does not come with a hardware random number generator, we had to improvise and use other sources of entropy to generate random values.

- On initialization, we stream read all of SRAM (yes, all 32768 bytes) and continuously update a hash value.
- On initialization, we perform 1024 samples of the internal CPU temperature sensor and generate a hash value.
- On initialization, we get 128 samples of the internal tick timer and generate a hash value. (Subtle differences in the tick timer will cause a significantly different hash value).
- After we collect these hash values, we combine them all (XOR) to create a 32-bit entropy seed, which we use to initialize a PRNG function.
- In addition to this initial system entropy, we combine the generated PRNG value with the tick timer value on a user-event, specifically when the car receives an unlock request. Because the tick timer is so precise (on the manner of microseconds), it is practically infeasible to replicate an unlock request at the exact tick timer. Combine this with controlling the other sources of entropy on system initialization and it becomes an impossible feat to generate a replicated nonce.

Security Requirements

SR1

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

- The car verifies the signature of the fob thus ensuring it is an authentic fob.
- A fob cannot obtain the private key to unlock the car without being properly paired or being a paired fob packaged by the manufacturer.
- It is important that we do not leak the private key through implementation flaws.

- It is important that we have a secure signing method so we do not leak the private key.

SR2

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

- The signed nonce challenge-response system enforces that a fob has to be used to provide a valid response.

SR3

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

- The nonce is randomly generated by the car on every request. Considering the nonce is 64 bits, it is extremely unlikely that a nonce will be reused for a replay attack.

SR4

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

- With only the PIN and no paired fob, the attacker does not have access to a private key from a paired fob and therefore has no way to create a paired fob.
- With a paired fob and an invalid PIN, pairing process not work and secrets won’t be transferred to the new fob.

SR5

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

- The manufacturer private key is only ever generated in the deployment environment and is not stored anywhere on the devices.
- Only the manufacturer private key can sign valid feature packages.
- The car never modifies its EEPROM state and is read only, including the manufacturer public key.

SR6

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

- Manufacturer key pair signatures with unique car ID prevent an attacker from forging or reusing a packaged feature.