2024 Design Document
Secure MISC

0xDACC

April 2, 2024

1   Proposed List Changes

Use standard I2C packet structure

Header		Payload
Packet Magic	Checksum	Length
(1 byte)	(4 bytes)	(1 byte)
		0x00

Table 1: List Packet

Header		Payload	
Packet Magic	Checksum	Length	Data
(1 byte)	(4 bytes)	(1 byte)	(4 bytes)
		0x04	

Table 2: List Response Packet

2   Proposed Attest Changes

Store attestation PIN as a hash with enough rounds that it takes approximately 2 seconds.

- Limits brute force attempts

- Makes raw PIN unable to be extracted from flash

Wrap attestation symmetric key with attestation PIN hash

Store attestation data encrypted with unwrapped symmetric key

- Also limits brute force and makes PIN unreadable from flash

Meets SR3 and SR4

Header		Payload		
Packet Magic	Checksum	Length	Data	Signature
(1 byte)	(4 bytes)	(1 byte)	(6 bytes)	(64 bytes)
		0x06	0x415454455354	

Table 3: Attestation Data Packet

Header		Payload		
Packet Magic	Checksum	Length	Attestation Data	Signature
(1 byte)	(4 bytes)	(1 byte)	(192 bytes)	(64 bytes)
		0xC0		

Table 4: Attestation ACK Packet

3 Proposed Replace Changes

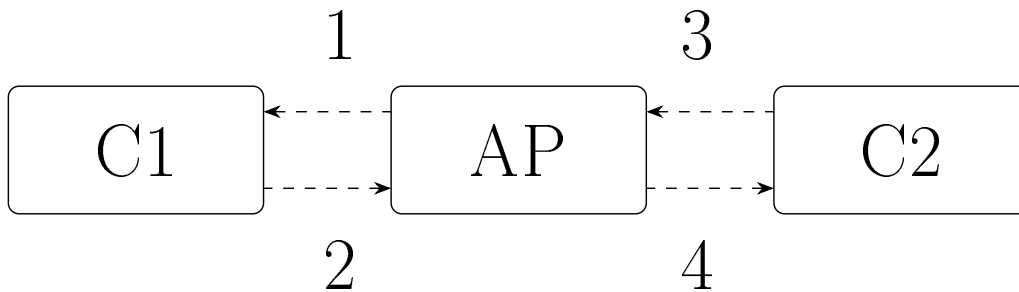**Store replacement token as a hash**

- Makes token unable to be extracted from flash

- Highly unlikely that the token can be brute forced

Meets SR3

4 Proposed Boot Changes

Verify integrity of all 3 boards

- Store public key C and private key A on AP

- Store public key A and private key C on Components

1. AP verifies Component1

    (a) AP generates a random number and asks Component1 to sign with key C

    (b) AP verifies signature using key C

2. Component1 verifies AP

    (a) Component1 generates a random number and asks AP to sign with key C

    (b) Component1 verifies signature using key C

    (c) Component1 boots

3. AP verifies Component2

    (a) AP generates a random number and asks Component2 to sign with key C

(b) AP verifies signature using key C

4. Component2 verifies AP

   (a) Component2 generates a random number and asks AP to sign with key A

   (b) Component2 verifies signature using key A

   (c) Component2 boots

5. AP verifies ComponentN

   (a) AP generates a random number and asks ComponentN to sign with key C

   (b) AP verifies signature using key C

6. ComponentN verifies AP

   (a) ComponentN generates a random number and asks AP to sign with key A

   (b) ComponentN verifies signature using key A

   (c) ComponentN boots

7. AP Boots

If any signatures are invalid, stop immediately and shut down.

If:

- Packet Magic != Expected Magic

- CSUM(Payload) != Expected Checksum

- Length != 0x40

- ecrecover(signature) != key A or key C

Shut down immediately, send fail packet if running on component, and do not continue operation.
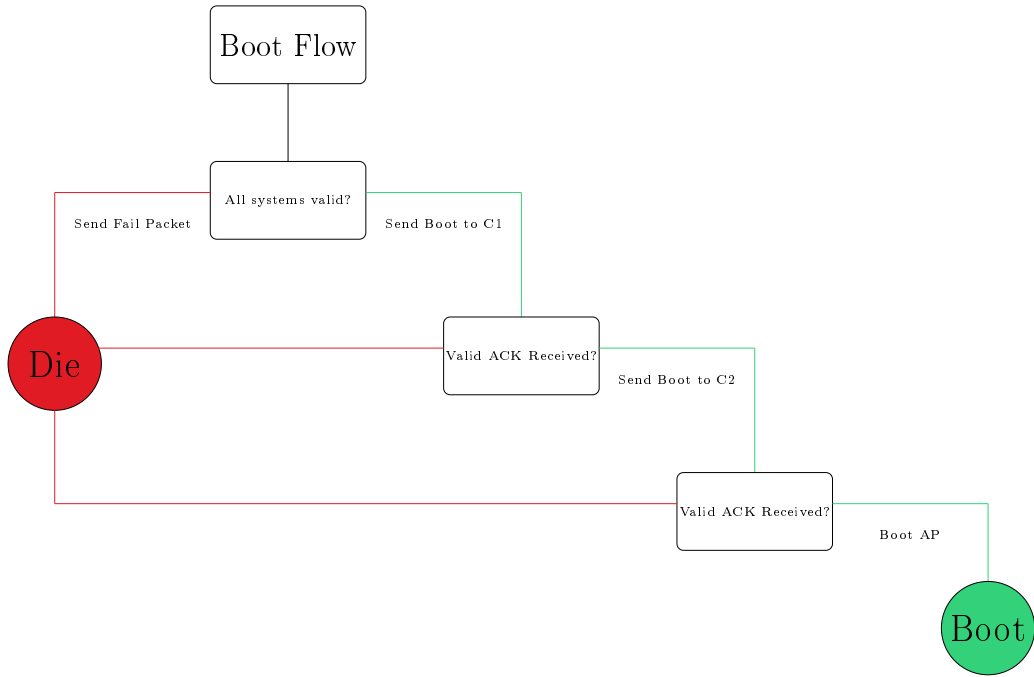
Meets SR1 and SR2

Header		Payload		
Packet Magic	Checksum	Length	Data	Signature
(1 byte)	(4 bytes)	(1 byte)	(32 bytes)	(64 bytes)
		0x60		

Table 5: Component Boot Packet

Header		Payload		
Packet Magic	Checksum	Length	Boot Message	Signature
(1 byte)	(4 bytes)	(1 byte)	(64 bytes)	(64 bytes)
		0x40		

Table 6: Boot ACK Packet

5 Proposed Secure TX Changes

ECIES Based Scheme

- Generate private key using RNG

- Create an encrypted channel even though unnecessary.

- Confidentiality will be provided to make RE'ing just a tiny bit harder

- Encrypt packets with negotiated key

- Negotiate HMAC key over new channel

- Append HMAC to all packets before encrypting

- Calculate checksum of encrypted data

If:

- Packet Magic != Expected Magic

- CSUM(packet) != Expected Checksum

- Payload Magic != Expected Magic

- HMAC(Data) != HMAC or Hash(Key) != Key Hash

- Nonce != Expected Nonce

Shut down immediately, send fail packet, and do not continue operation.

Meets SR5

Header		Encrypted Payload				
Packet Magic	Checksum	Payload Magic	Length	Nonce	Data	HMAC
(1 byte)	(4 bytes)	(1 byte)	(1 byte)	(4 bytes)	(256 bytes)	(32 bytes)
		0xDD				

Table 7: Encrypted I2C Packet

Header		Payload	
Packet Magic	Checksum	Length	Key Material
(1 byte)	(4 bytes)	(1 byte)	(64 bytes)
		0x40	

Table 8: Key Exchange I2C Packet

6 Summary

6.1 SR1 All components must be valid for AP to boot

- Validate all component's integrity through signing an arbitrary number

- Components then validate the AP to make sure all systems are present and valid

- Boot the AP

6.2 SR2 All components must be validated by AP and commanded before booting

- After a successful handshake, it can be assumed that all components are valid

- Send signed boot command to components from AP

- Boot individual components

6.3 SR3 The Attestation PIN and Replacement Token should be kept confidential

- PIN will be stored as a hash with enough iterations to reduce the brute force likelihood

- Replacement Token will also be stored as a hash

6.4 SR4 Component Attestation Data should be kept confidential

- Attestation Data will be stored with symmteric encryption with the key being derived from the Attestation PIN

6.5 SR5 Integrity and Authentication of all communications

- All messages will follow a standard packet format with a negotiated HMAC key and assymetric encryption

- A nonce and ephermeral keys may be included to limit replay attacks