FOUNDATION

PASSPORT SILICON SECURITY

How Passport Uses Silicon Features to Secure Your Bitcoin

Ken Carpenter
CTO, Foundation Devices

WHAT IS PASSPORT?

- Foundation's next-generation Bitcoin-only hardware wallet
- Fully air-gapped, with no wired or wireless communication
  - QR codes (Passport has a color camera)
  - microSD
- Improvements over the previous Passport Founder's Edition include:
  - Color screen with ultra-hard, scratch resistant glass
  - Removable lithium ion battery
  - USB-C connector with power only pins to preserve the air gap
  - Slimmer design incorporating elements of Art Deco

PASSPORT SECURITY PHILOSOPHY (1 of 2)

- Be as transparent as possible, with open source software, circuit designs, schematics, and bill of materials
- Only purchase components through reputable suppliers and distributors, usually US-based companies
- Make use of generally available components, so that experts can build their own Passports from scratch without our knowledge or permission

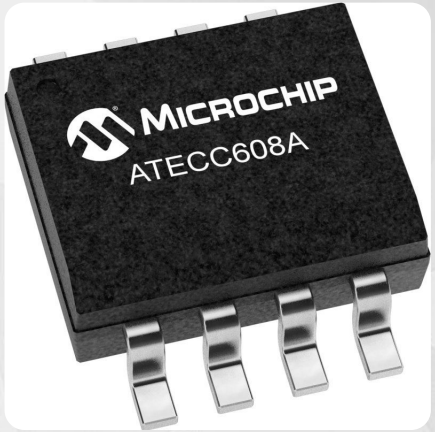
PASSPORT SECURITY PHILOSOPHY (2 of 2)

- Minimize the use of black-box silicon (*i.e., chips running unknown firmware*)
  - Prefer “dumb” chips that are not executing unknown code
- Assemble our hardware in the US, including circuit boards

PASSPORT SILICON

Passport's security is based around the interaction between two silicon components

STM32H753 MCU



ATECC608A Secure Element

The rest of this talk will cover how these components are configured and used in Passport

BUT FIRST, A BRIEF INTERLUDE ABOUT SUPPLY CHAIN SECURITY...

SUPPLY CHAIN SECURITY

- Supply Chain security for Passport encompasses everything from when the components are made, to proof of custody along the way, to who's on the manufacturing floor, to when the customer first receives the device
- Security is a concern at each step along the way
- But this conference is about silicon, so we'll focus on what happens after the chips arrive at our factory...

PCB ASSEMBLY

- PCBs go through the pick-and-place machine to assemble the parts onto the board, including the MCU and Secure Element
- Then the PCBs are sent for initial testing and provisioning
- We developed a custom "bed of nails" fixture that allows us to instantly connect dozens of GPIO pins to the production PCBs and read voltages and values from test points on each PCB
- This fixture also allows us to program the bootloader and firmware for each PCB
- We developed a custom testing and provisioning tool in Rust to control this fixture

FACTORY PROVISIONING

The provisioning tool performs the following steps for each PCB:

- Load a test bootloader and run through a series of hardware tests, stopping if anything is wrong and reporting diagnostic info to the operator
  - Screen controller, keypad, Secure Element, EEPROM, fuel gauge, system voltage levels, etc.
- Copy the Supply Chain secret to Passport (details later)
- Flash the normal bootloader & firmware, then restart Passport
  - During first boot, Passport configures the MCU and Secure Element

PASSPORT FIRST BOOT CONFIG (1 of 2)

MCU

- Bootloader checks to see if the MCU secrets have been configured
- If they have, then skip ahead to the normal bootloader processing
- If not, then we initialize them as described on the following slide

Secure Element

- Bootloader checks to see if the Secure Element's slot configuration and data values have been initialized
- If they have, then skip ahead to the normal bootloader processing
- We'll cover how each slot is used in detail shortly

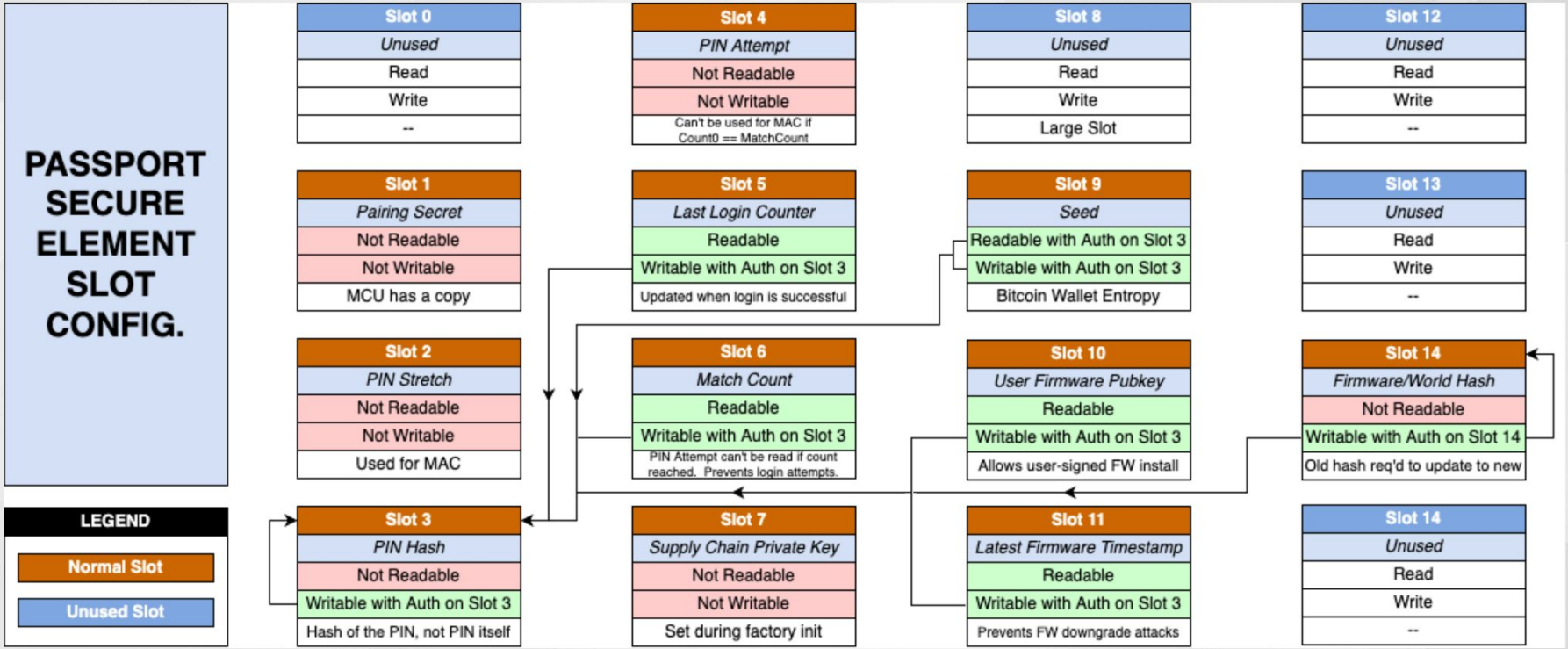
PASSPORT FIRST BOOT CONFIG (2 of 2)

MCU Configuration Sequence

Generate random secrets for:

- A One-Time Pad (OTP)
  - Used for hashing and XOR-encrypting the PIN
- Pairing Secret
  - Used to authenticate communication with the Secure Element (also saved to the SE later)
- Transaction cache encryption key
  - Passport caches some state from previously signed transactions to mitigate certain software wallet attacks that attempt to trick the user into re-signing similar transactions
  - This key is used to encrypt that data
- All of these MCU secret values are saved into the internal bootloader flash

SECURE ELEMENT SLOT CONFIG



FOUNDATION

SLOT USE DETAILS

The following slides will cover how Passport uses each slot in more detail

SLOT 1 - PAIRING SECRET

- To access most of the other slots in the SE, we have to perform a back and forth HMAC authorization process with the SE
- The purpose is to prove that the MCU knows the same secret value as the SE -- ***without this secret, access to the SE is impossible***
- The Pairing Secret is also stored in the internal flash of the MCU
- Once complete, we can proceed to read or write the desired slot
- This authorization process is temporary, and has to be repeated for ***every*** read or write

SLOT 2 - PIN STRETCH

- Contains a random value which is unknown to the MCU, and which cannot be read or written after provisioning
- Used for "PIN stretching", in which we repeatedly have the SE perform hash on the PIN with this secret
- Main purpose is to make PIN attempts take longer to add some additional protection against brute force attacks

SLOT 3 - PIN HASH

- Contains a heavily hashed version of the user's PIN
  - Hashed with both the PIN Stretch secret (many times) and the PIN Attempt secret (just once)
- The PIN hash is also encrypted by XORing it with the One-Time Pad (OTP) stored in the MCU
  - This means an attacker who manages to decap the SE and extract data from it still has to compromise the MCU and extract the OTP in order to decrypt the PIN
- Knowledge of this value is required to login to Passport

SLOT 4 - PIN ATTEMPT

- Contains a secret random value that is hashed with the user's PIN for each login attempt
  - This is separate from the PIN Stretch value
- This slot is used in conjunction with ***Slot 6 - Match Count***, and the monotonic counter
  - If the `monotonic counter == Match Count` , then all PIN attempts have been exhausted, and the SE will disallow access to ***Slot 4***
  - Without the ability to hash the PIN with this slot's value, it's impossible to login to Passport

SLOT 5 - LAST LOGIN COUNTER

- Contains a copy of the monotonic counter's value the last time the user successfully logged in

- Used to keep track of the number of failed attempts to display to the user

SLOT 6 - MATCH COUNT

- Contains a value that is compared against the monotonic counter internally by the SE
- If the `monotonic counter == Match Count`, then all PIN attempts have been exhausted, and the SE will disallow access to ***Slot 4***, thereby making login impossible
- The SE requires that the Match Count be a multiple of 32
  - This, unfortunately, results in some additional incrementing of the monotonic counter after a successful login

SLOT 7 - SUPPLY CHAIN PRIVATE KEY

- Contains a secret key written to each Passport at the factory
- Used in a challenge-response HMAC process with the Foundation validation server to prove that a Passport is genuine
- A separate slide on the Supply Chain Validation process is coming later

SLOT 9 - SEED ENTROPY

- Contains the entropy bytes for the user's Bitcoin seed
- These are the byte values used to lookup words in a BIP39 seed word list
- The seed is XOR'd with the OTP from the MCU before storing it in the SE
    - An attacker who manages to decap the SE and extract this slot's value still has to compromise the MCU and extract the OTP in order to decrypt the seed

SLOT 10 - USER FIRMWARE PUBKEY

- Contains a public key which the user has decided to trust to sign firmware images
- This is a feature for advanced users or developers which allows them to load their own firmware
- The bootloader will only install and load firmware that is either signed by Foundation or signed by the public key in this slot
- Only a logged-in user is able to install a User Firmware PubKey

SLOT 11 - LATEST FIRMWARE TIMESTAMP

- Contains the Linux timestamp of the most recently installed firmware that was officially signed by Foundation
- Does **NOT** get updated when installing custom user builds or Foundation beta releases
- The bootloader will not install officially signed firmware that is older than this timestamp
- This prevents downgrade attacks

SLOT 14 - FIRMWARE WORLD HASH

- Contains a hash which combines the current Firmware Hash with the Device Hash, which consists of:
  - SE Serial Number
  - MCU OTP
  - Pairing Secret
  - MCU Unique ID
- This slot is updated whenever a valid signed firmware image is installed (both officially-signed and user-signed)
- The SE requires proof of this value at boot to turn the security LED from red to blue

SUPPLY CHAIN VALIDATION

- Passport uses ***Slot 7 - Supply Chain Private Key*** to help validate that Passport was not modified after leaving our factory

- We provide two methods for validation
    - **Manual** - The user visits the validation website directly and interacts with Passport to complete the validation process
    - **Envoy** - Use our mobile app to facilitate the communication between Passport and the Foundation validation server -- ***this is the preferred method***

MANUAL SUPPLY CHAIN VALIDATION

- User navigates to `validate.foundationdevices.com` on their phone or computer to display a QR code challenge
  - Challenge is a random value hashed with the value of ***Slot 7 - Supply Chain Private Key***
  - The challenge is signed by a private key owned by the server
  - Passport knows the public key

- User scans the QR code with Passport
  - Checks that the challenge is signed by the validation server private key
  - Asks the SE to perform an HMAC with the challenge value and the value in Slot 7
  - Converts the result to a set of 4 words, which are displayed to the user

- User enters the 4 words into the Foundation validation website and is presented with a Passed or Failed result

ENVOY SUPPLY CHAIN VALIDATION

- While adding a new Passport in Envoy, Envoy will fetch a random, signed challenge from the Foundation validation server and present it as a QR code

- User scans the challenge QR code with Passport
  - Checks that the challenge is signed by the validation server private key
  - Uses the SE to perform an HMAC with the challenge value and the value in Slot 7
  - Converts the result to a response QR code, displayed on Passport

- User scans the response QR code with Envoy
  - Envoy sends the response data to the Foundation validation server and receives a pass or fail result
  - This result is displayed to the user

OTHER SECURITY FEATURES

Passport incorporates many other security features in both the bootloader and the main firmware.

The file `SECURITY.md` on our GitHub goes into more detail on most of them.

https://github.com/Foundation-Devices/passport2/SECURITY/SECURITY.md

FOUNDATION

FUTURE IMPROVEMENTS (1 of 2)

1. Further encrypt ***Slot 9 - Secret*** with an alternate PIN hash
   (not the same PIN hash that is stored in the ***Slot 3 - PIN Hash***)
   - This will add some significant extra protection against compromise of the SE
   - Attackers would need to brute force the PIN to decrypt the secret, and every possible PIN results in a seed that they would need to check to see if there is any Bitcoin on the blockchain at that seed

2. Incorporate PKI features of the SE and Microchip library
   - One use case of this particular SE is to prove that a part which contains it is genuine using PKI
   - Microchip provides support for this sort of PKI use case, and we are considering using it in a future device (by design, we can't change the slot configuration of shipped devices)

FUTURE IMPROVEMENTS (2 of 2)

4. Upgrade from the Microchip ATECC608a to the ATECC608b

- There are some minor improvements to the 'b' variant, but the level of improvement is largely unknown, as Microchip won't say what is different between the two versions
  - Some operations are significantly slower with the new variant
  - Other than timing it is a pin-compatible upgrade

- This is mostly a market perception concern, as the security of the ATECC608a is already very high, especially compared with other hardware wallets that don't employ a Secure Element at all

SILICON DREAMS (1 of 2)

What would we like in an ideal future SE?

1. Much larger key store to support multiple root seeds

2. Ability to run common crypto functions directly on the chip
   - Bitcoin transaction signing (ECDSA, Schnorr)
   - Support for large transactions
   - Secrets never need to leave the SE!

3. Have MCU and SE be on the same die or even just the same package, but still be separate processing domains
   - Removes possibility of spying on traffic (although that is encrypted now anyway)
   - Makes for a single BoM item

SILICON DREAMS (2 of 2)

What would we like in an ideal future SE?

4. More monotonic counters to support multiple PINs (only one on our SE today)

5. Configurable rate limiting to control frequency of PIN attempts at the hardware level

6. Developer SKU of the SE that allows reading internal state
    - Useful for initial driver development

FOUNDATION

QUESTIONS?