MA35D0 Secure Boot

Introduction

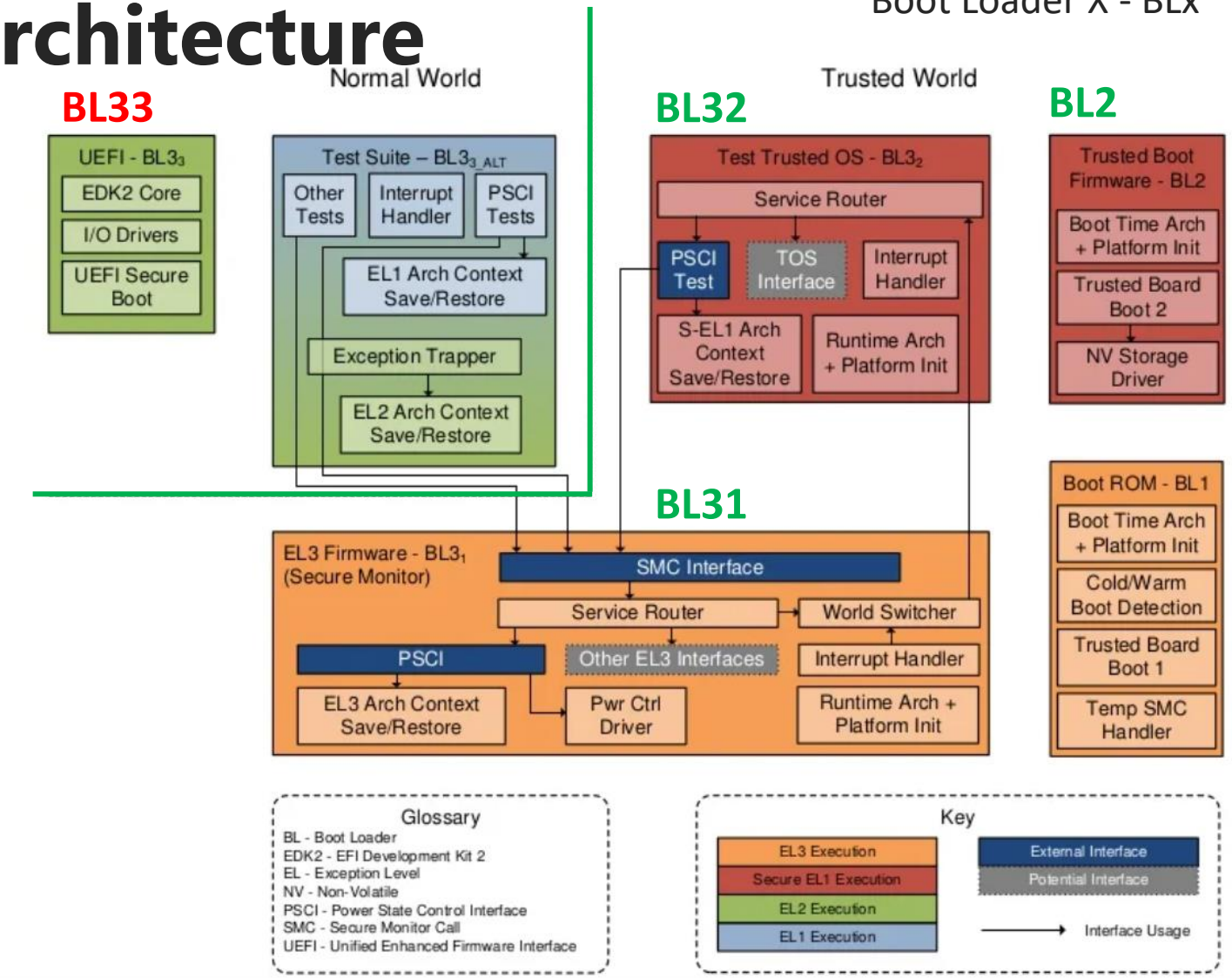
Aaron Chen

MA35D0 TF-A FW Architecture

- Reference implementation of secure world software
- Boot loader in each stage

Secure

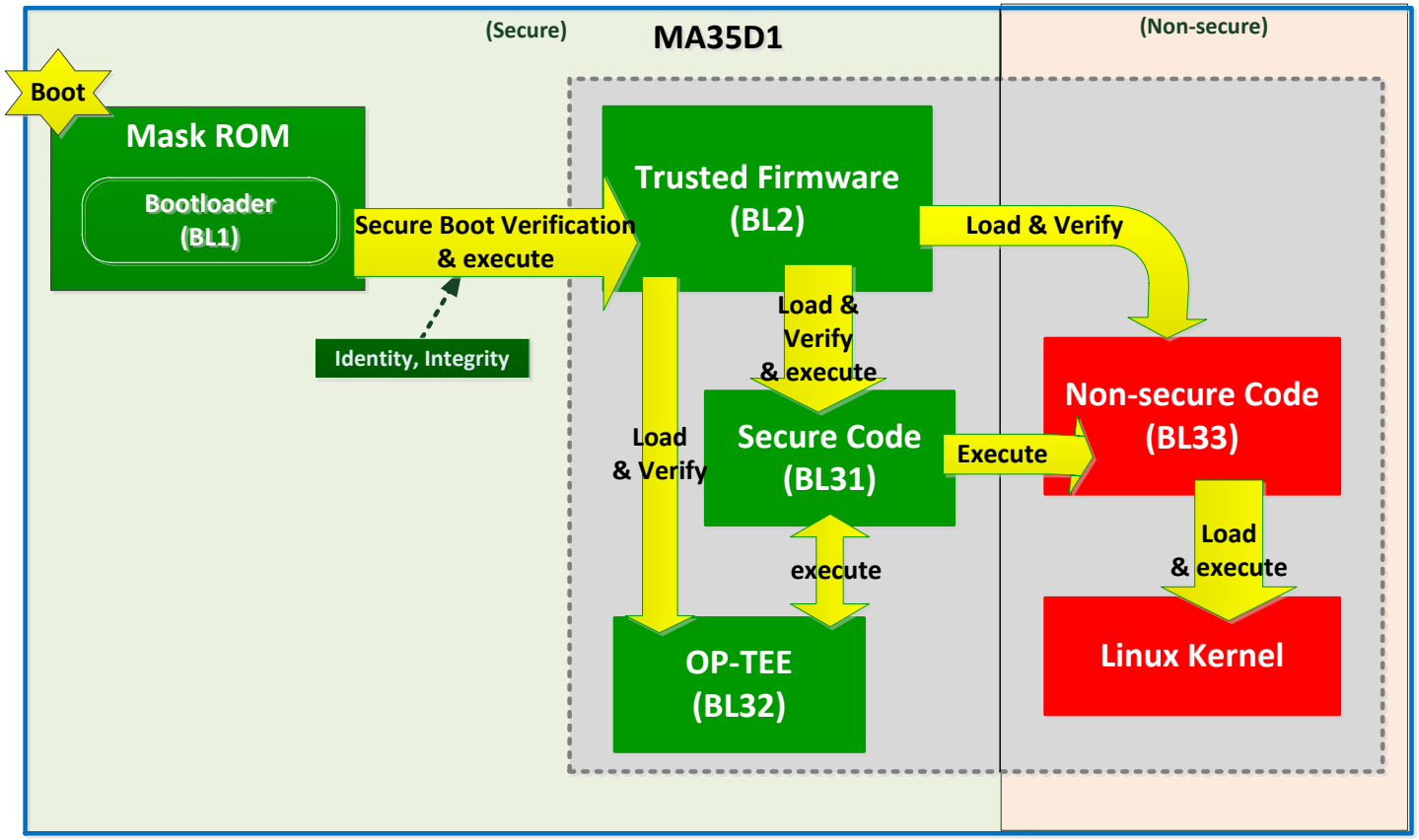
- Trusted Board Boot Requirement- TBBR- **BL2**
- Power Status Coordination Interface and Secure monitor - **BL31**
- TEE OS – **BL32 (OP-TEE)**
- U-boot – **BL33**



Source: https://firmwaresecurity.com/2015/05/23/arm-trusted-firmware/

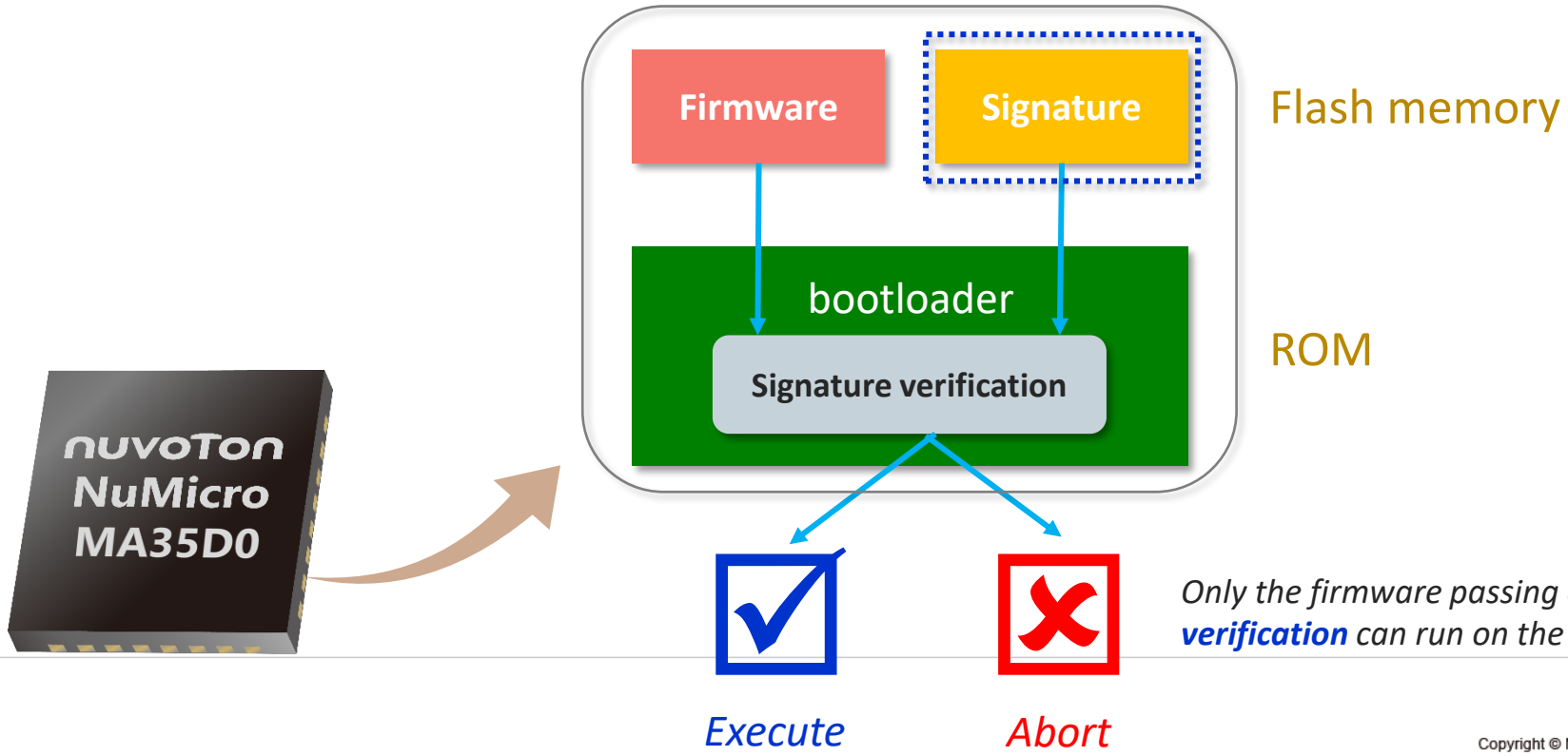
MA35D0 Boot Sequence with TF-A

- Root of trust by Mask ROM(IBR)
- Implemented using ECDSA and AES
- First stage
  - IBR(BL1) -> BL2
- Second stage
  - BL2 -> BL31 -> BL32 -> BL33



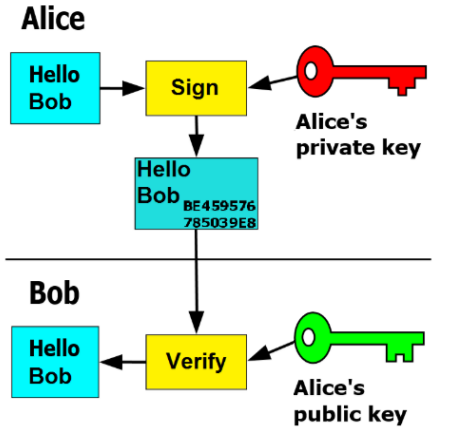
MA35D0 Firmware Authentication (1/3)

- **Secure Boot** authenticates application code before allow it to run.
  - immutable ROM code verifies firmware's digital signature after system power-on



MA35D0 Firmware Authentication (2/3)

- **Digital Signature**
  - Digital signatures employ asymmetric cryptography
    - ECDSA (Elliptic Curve Digital Signature Algorithm)
    - RSA PKCS#1 and ANS X9.31
  - Provides **Data Integrity** and **Non-repudiation** (**Authenticity**)
  - It is a **cryptographic value** that is calculated from the data and a private key known only by the signer.
    - Security level depends on the size of key (usually expressed in "bits“)



Source : WIKIPEDIA

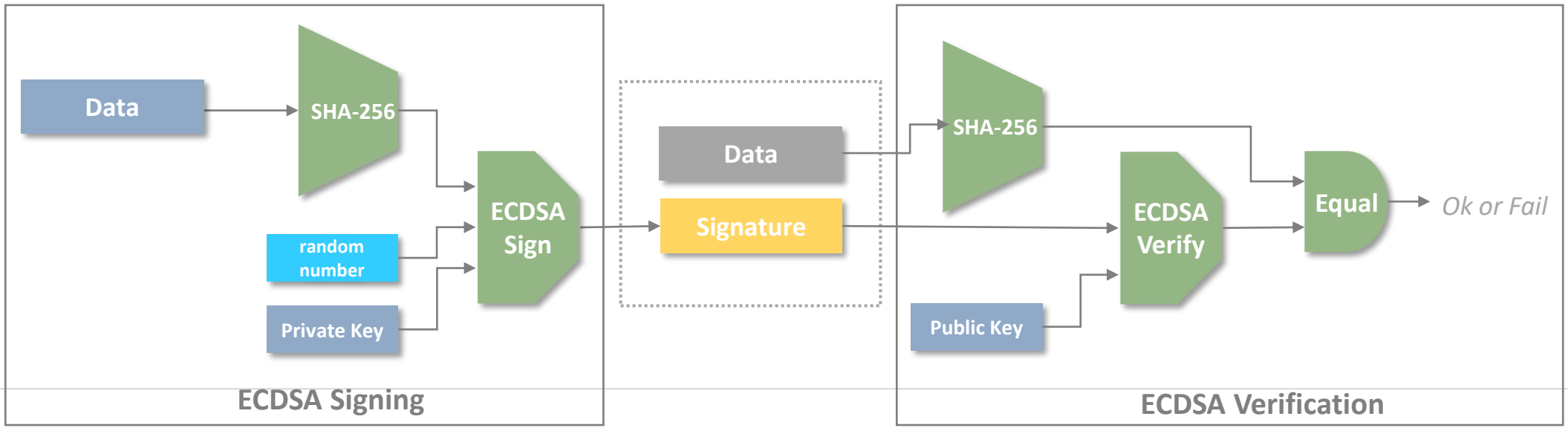
Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

SHA-256 : has 128-bit security level to against collision attacks

MA35D0 Firmware Authentication (3/3)

- Digital Signature – cont.
  - Digital signature algorithm used in secure bootloader : **ECDSA P-256**
    - Private key (256 bits)
    - Public key (256 + 256 bits)
    - Signature (256 + 256 bits)



MA35D0 Key Store

- **Key Store OTP**
  - Support store key in Key Store OTP
  - Store 1 AES key and 1 ECC public key pair for secure boot (key number: 6-8)
  - Key Store can be set in the secure region and access only by secure OS, Linux can not accessible
  - Secure boot key is set “not readable for CPU”

OTP Key Number	Key Naming	Key Storage Space	Erase Time
0	Hardware Unique Key (HUK)	128 bits	1
1	Hardware Unique Key (HUK)	128 bits	1
2	Hardware Unique Key (HUK)	128 bits	1
3	User defined key	256 bits	1
4	User defined key	256 bits	1
5	User defined key	256 bits	1
6	IBR ECC public key	256 bits	1
7	IBR ECC public key	256 bits	1
8	IBR AES key	256 bits	1

MA35D0 Secure boot with Yocto

- **Set the AES and ECC keys before build up image**
  - Yocto: /yocto/sources/meta-ma35d0/conf/machine/numaker-iot-ma35d06f80.conf
  - $ bitbake core-image-minimal -c cleanall
  - $ bitbake core-image-minimal

```
# ================================================================
# secure boot
# ================================================================
# "yes" boot from secure mode
# "no" boot from normal mode
SECURE_BOOT = "yes"
# The NuWriter will encrypt all relative files by AES,
# and calculate the ECDSA signature
AES_KEY = "0A0BC81E5AFBF836C5E0CFBEA12C1E269A2EBC3B0B6EC39EEE1C7142F760EBC4"
ECDSA_KEY = "8F1571E9975006A545DF854264F7B18934B5CB2EE619E66DBC0FBEA52C71C919"
```

Private key

MA35D0 Secure boot with Buildroot

- **Buildroot**
  - Write the key into config file through “make menuconfig” command.

```
Bootloaders  --->
  [*] MA35D0 Secure Boot
  (0A0BC81E5AFBF836C5E0CFBEA12C1E269A2EBC3B0B6EC39EEE1C7142F760EBC4) AES Key
  (8F1571E9975006A545DF854264F7B18934B5CB2EE619E66DBC0FBEA52C71C919) ECDSA Key
$ make
```

Private key

MA35D0 Secure boot key program

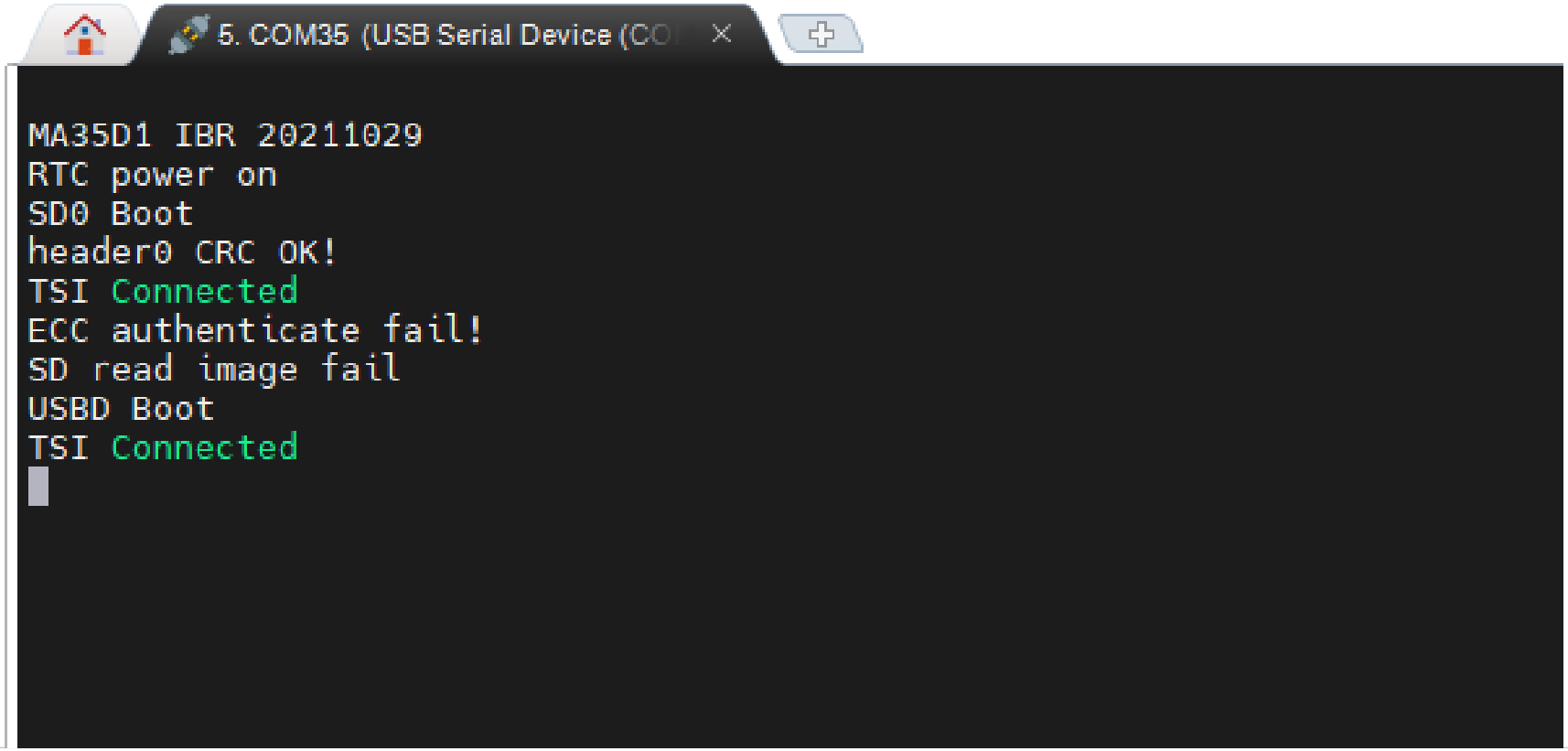
- **Program aes and ecc public Keys into Key Store IBR region**
  - The NuWriter supports to program the keys into MA35D0 Key Store
  - Use Prepare “key.json” to program the key : $ **nuwriter.py –w otp key.json**

key.json

```
{
   "publicx": "72F84F681092E3A05C1437E3E40534962A5C70556025D348FF9DB97D6AF83EB5",
   "publicy": "8D32DAC7AB6F90332E8E0060E159E0B31502BB4FB2D78369F02D1F5B0C335AD3",
   "aeskey" : "0A0BC81E5AFBF836C5E0CFBEA12C1E269A2EBC3B0B6EC39EEE1C7142F760EBC4"
}
```

MA35D0 Boot Sequence with TF-A

- Authenticate fail → run in while 1 loop



MA35D0 Secure boot – two level protection (1/2)

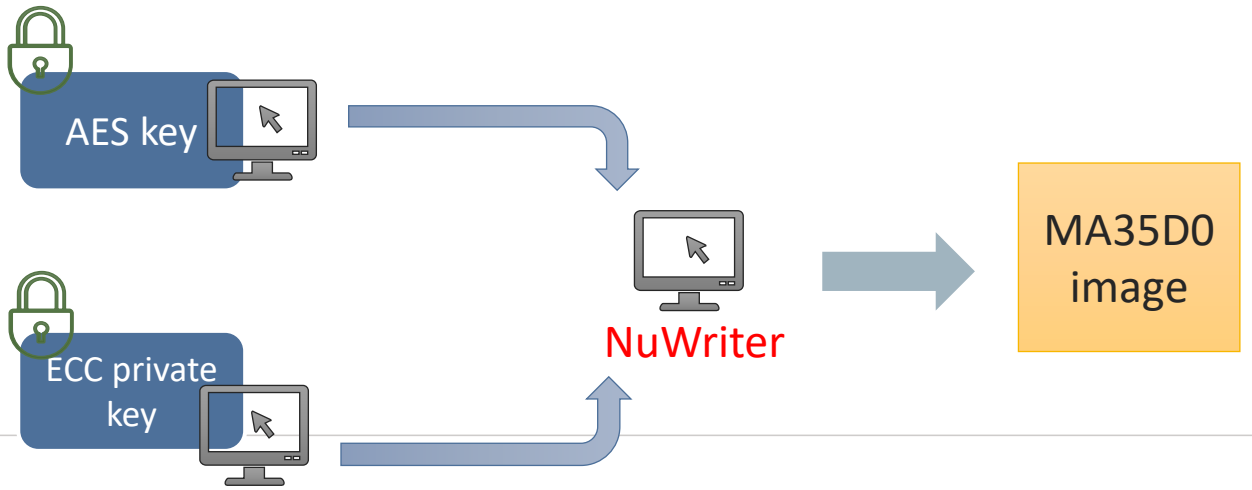
- **ECDSA with P-256 curve**
  - Use SHA256 and ECC to generate the signature of the image
  - Verify the image during the power on stage before AES decryption

- **AES-256 CFB mode**
  - Encrypt firmware image into Storage e.g. SD/eMMC, NAND, SPI NOR..
  - Decrypt firmware image after ECDSA pass and load image to the DDR

MA35D0 Secure boot – two level protection (2/2)

- **Stealing only one key cannot break the security boot mechanism.**
  - Only have ECC public key cannot encrypt and decrypt the image
  - Only have AES key cannot sign and verify the image

Storing ECC and AES keys with different methods enhances security.



Joy of innovation

NUVOTON

Thank You

Danke

Merci

ありがとう

Gracias

Kiitos

감사합니다

धन्यबाद

شكرا لك

הדות