硬體安全設計與操作.

2025/05/02

PUF security
AN ememory COMPANY

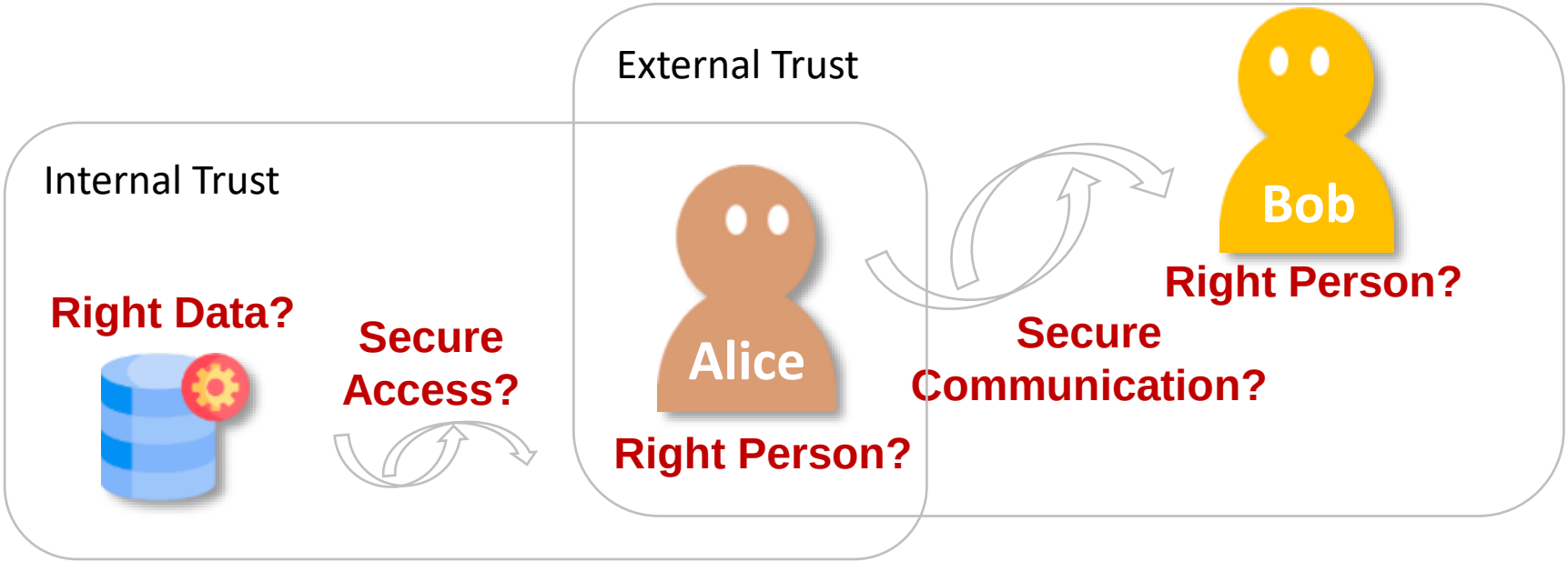
硬體安全設計與操作

1. Root-of-Trust (Recap)
2. HRoT Definition
3. Secure Primitives
4. Cryptos, Protocol and Certificate
5. Secure Boot

硬體安全設計與操作

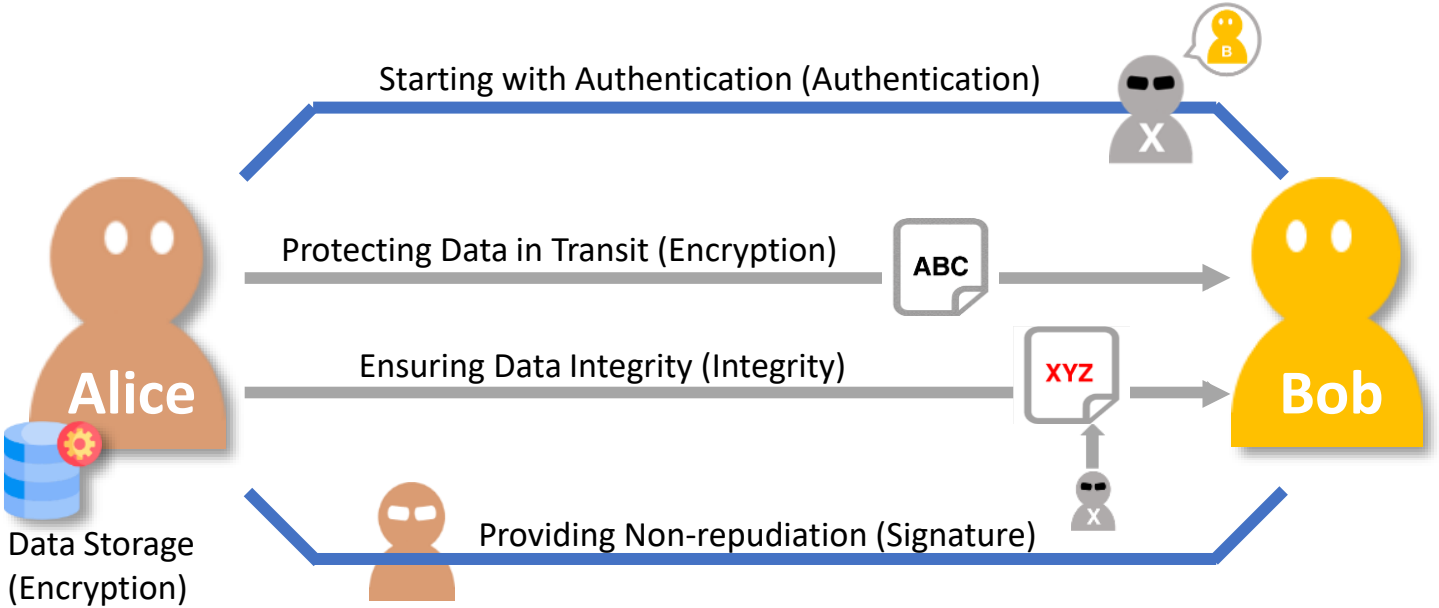
Hardware Root-of-Trust: Recap

RoT for Internal and External Trusted Basis .



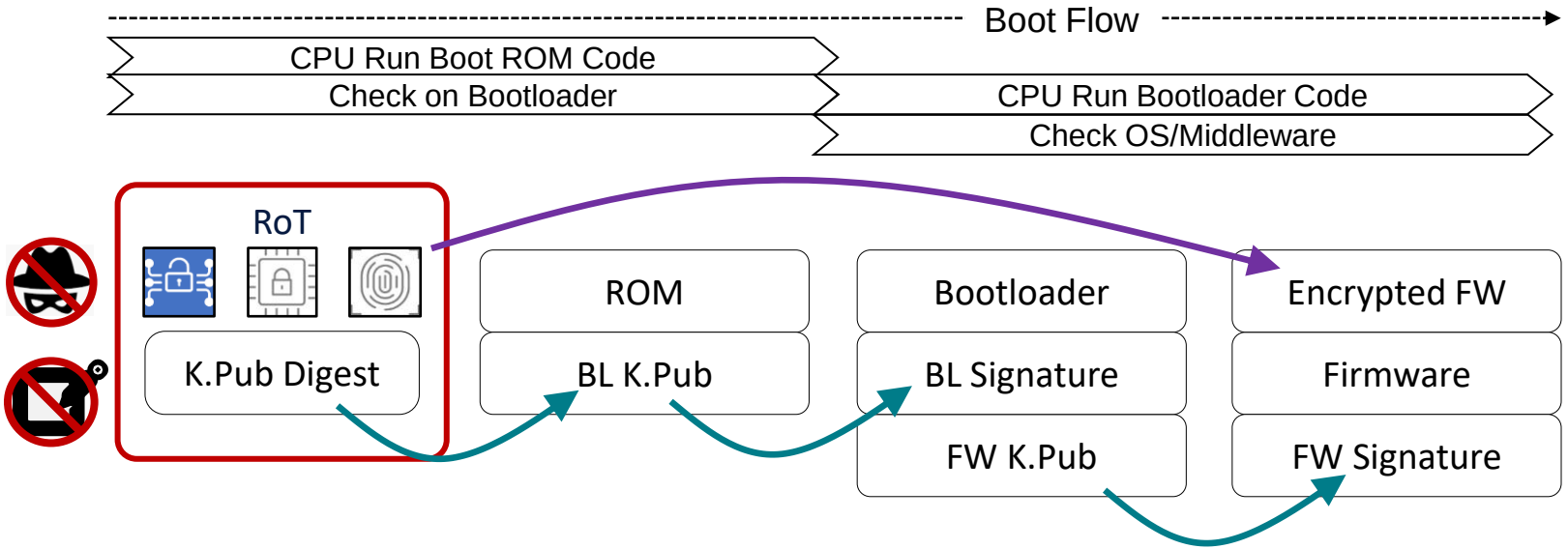
Root-of-Trust is the root of information. It should be trusted and free from doubt.

External Security: Secure Communication .



Security fundamentals: 1. trusted keys 2. certificate for authentication
3. digest for integrity check 4. cryptography for protecting the transmission

(SoC) Internal Security: Secure Boot .



HRoT provides **Secure Storage**, **HUK** and **Cryptos** for: 1. As anchor of chain of trust, 2. RSA/ECC and SHA for digest and verification 3. FW protection using AES with local PUF key

(SoC) Information that A RoT Should Protect .

	For External Security	For Internal Security
Keys	• UID • Root Key (HUK) • Private/Public Key • Shared Key • CA Public key (Digest) • Global FW Key	• Root Key (HUK) • Private/Public Key • FW Public Key • Local KEK/Local FW Key
Certificate	Service Certificate (X.509)	FW Signature (Secure Boot) Key Certificate (X.509)
Secure Info.	Chip Lock/Unlock PWD	Versioning/Debugging

Can't be stolen (Must Important): HUK(Private Key), KEK (Local Key) and FW Key

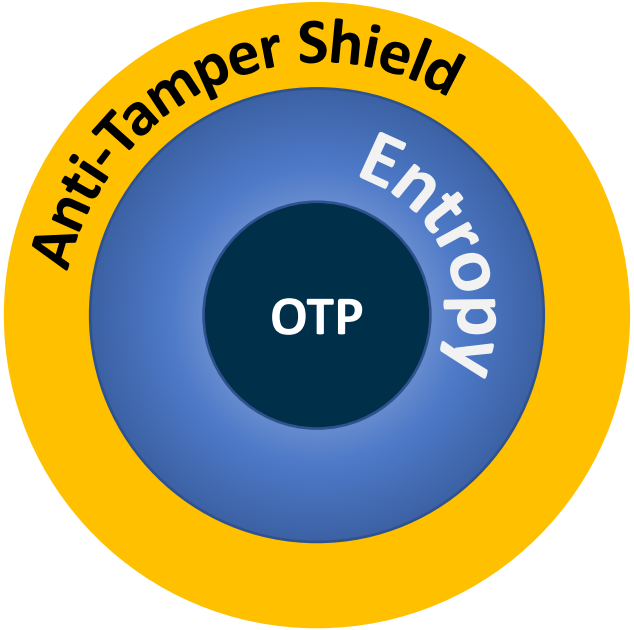
Can’t be revised (For Secure Boot, Authentication): CA Public key and Certificate.

→ HRoT is needed to robustly protect these secret keys and information than RoT

硬體安全設計與操作

HRoT Definition

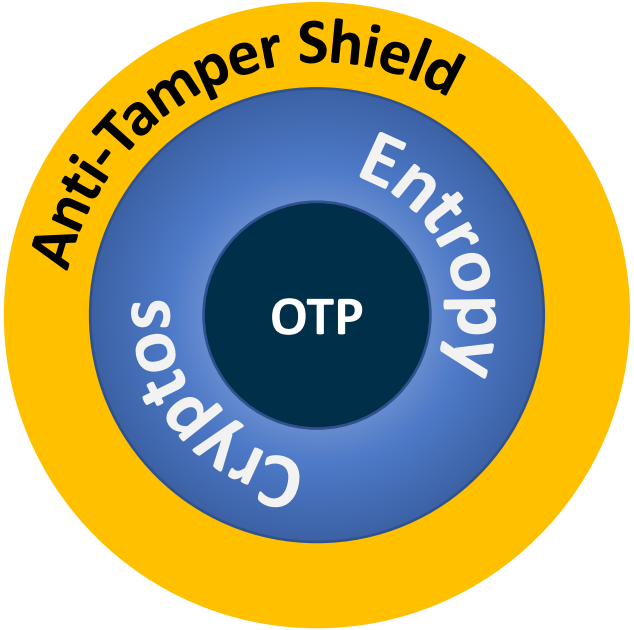
(Basic) Hardware Root of Trust .



Design Consists of
OTP
Entropy: PUF
Entropy: TRNG
Ant-tampering

Aim for
Key Generation
Key Storage
Secure Operation

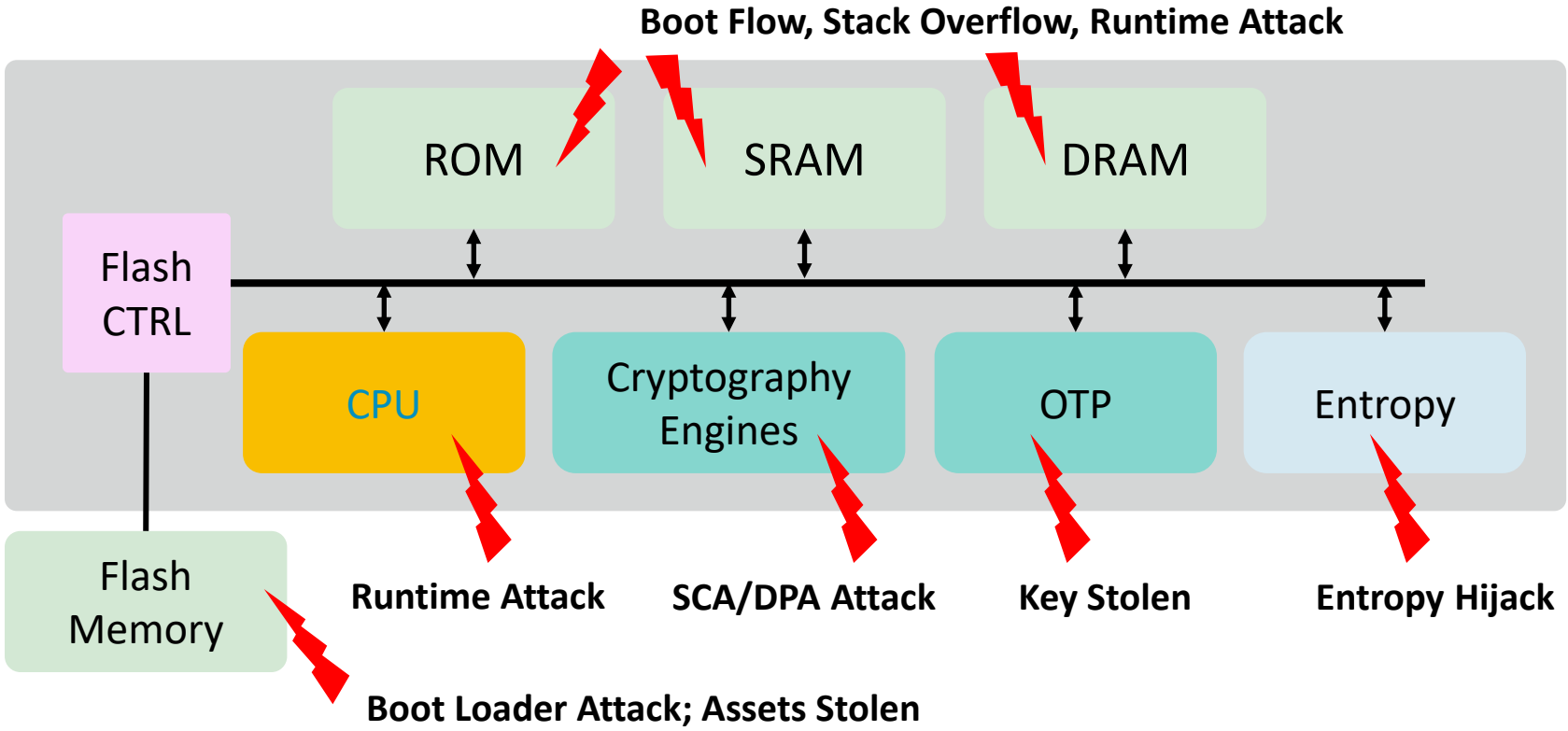
(Standard) Hardware Root of Trust .



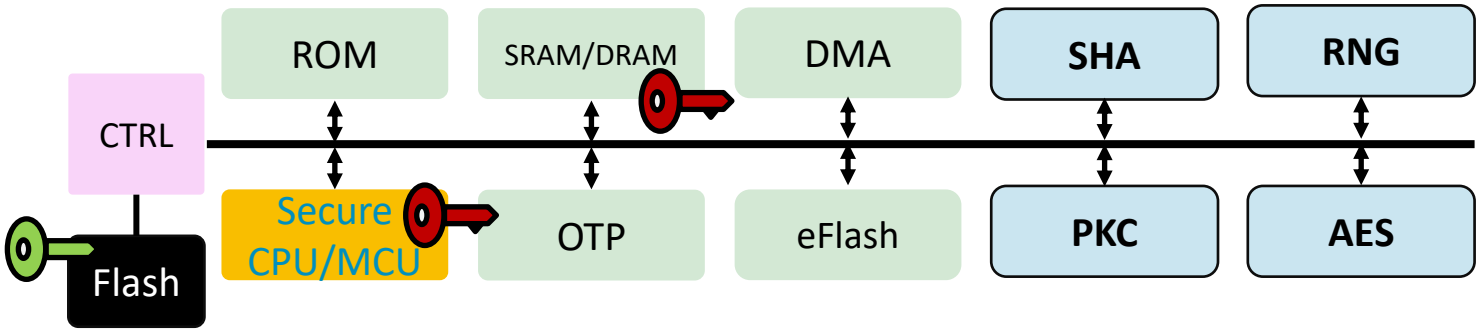
Design Consists of
OTP
Entropy: PUF
Entropy: TRNG
Cryptos
Ant-tampering

Aim for
Key Generation
Secure Storage
Secure Operation
Secure Environment

Vulnerability for a Convention SoC

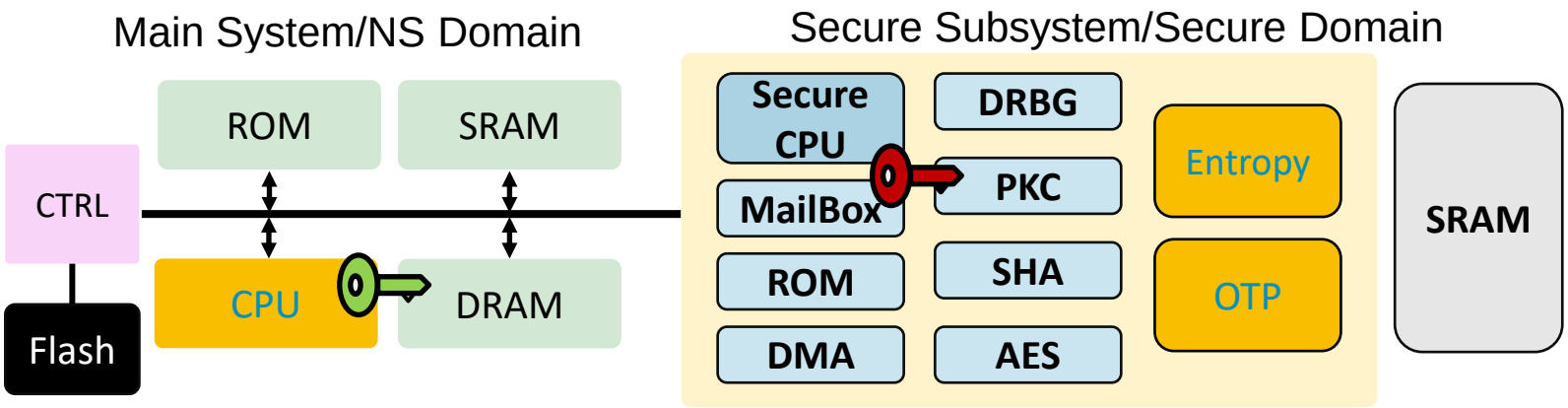


S1: Convention Secure SoC



- 最常見的系統安全設計架構
- 離散的密碼學硬體加速器提供加密運算
- 需要安全處理器以及各單元的抗攻擊設計

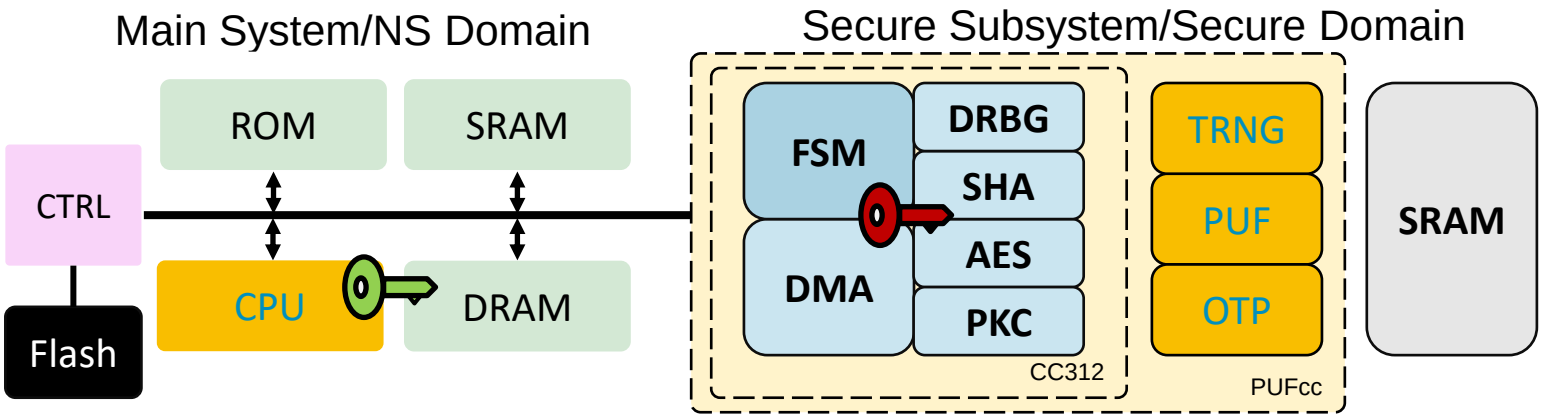
S2: Secure Subsystem using Secure CPU



降低被攻擊的缺口，隔離金鑰跟一般非安全資訊

- **設計獨立區塊，保護金鑰及重要系統資訊, 並永久隔離在安全邊界內**
- **主CPU/記憶體不能碰觸金鑰，金鑰外存都需要做KWP加密保護。**
- **主動的安全隔離區, 擁有OTP存儲、專用的工作記憶體和硬體加速單元。**

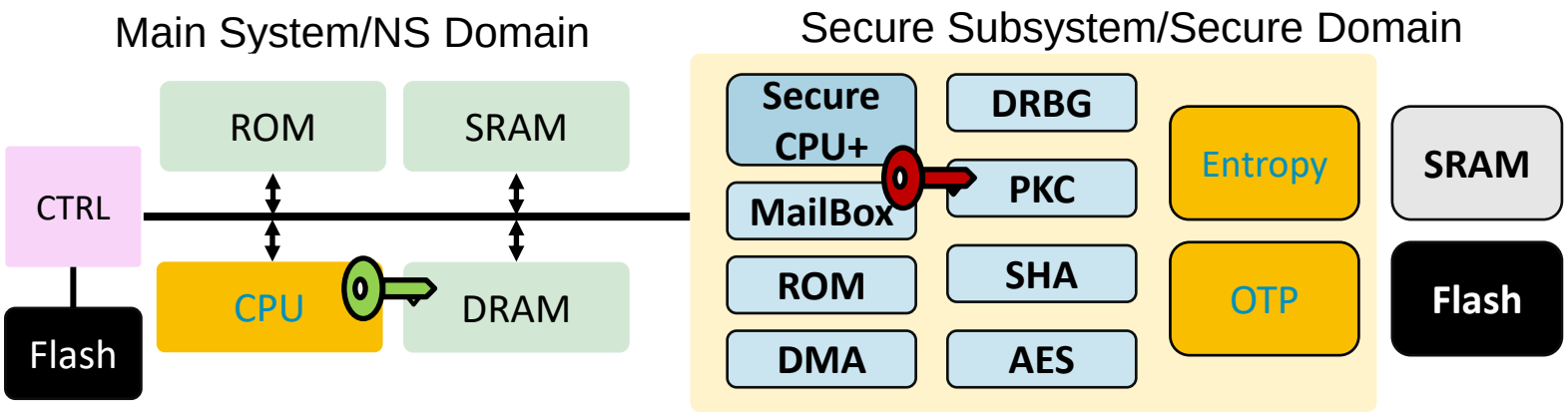
S3: Secure Subsystem using FSM



降低被攻擊的缺口，隔離金鑰跟一般非安全資訊

- **保護金鑰以及重要系統資訊於一層純硬體的安全邊界內。**
- **使用FSM (Finite State Machine) ，不使用CPU，更難被攻擊。**
- **被動的安全隔離區，如果有高階的安全應用(如付費)，需要額外CPU運算。**

S4: Secure Subsystem using Secure CPU+



在S2 的基礎之下，增加高階安全應用的使用場景。

- **完整執行安全應用、保護金鑰、憑證以及重要系統資訊, 並被隔離在安全邊界內**
- **主動的安全隔離區, 擁有安全的NVM存儲、專用的工作記憶體和硬體加速單元**
- **因為需要執行安全應用程式，需要增加ROM、大容量Flash、所以需要更強力的CPU/OS/FW, 以及相對應的 SoC I/F，複雜度大幅度提高。**

Comparison among HRoT

Solution	S1	S2	S3	S4
Secure IP	Discrete Cryptos	RT-120 (Rambus) eSecure (Secure IC) tRoot Vx (Synopsys)	CC312/CC712(ARM) PUFcc3	RT-650 (Rambus) tRoot Fx (Synopsys) RISC-V+/PUFcc3
2nd CPU/HW	--	• Secure CPU • ROM/RAM/(Flash)	--	• Secure CPU+ • ROM/RAM/Flash
Pros	• **Flexible cryptos**	• **Flexible cryptos** • **Active HRoT**	• **Cost Effective** • **Fixed cryptos** • **Highest Security**	• **Trusted Secure Applications** • **Asset Protection**
Cons	• Secure CPU • Key Protection	• Secure CPU is MUST • ROM and Flash are MUST for Booting	• Function limitation using FSM	• Complexity as a SoC • Secure CPU+, ROM and Flash are MUST

Ch-2 .

Secure Primitives in a HRoT

Components of the Hardware Root of Trust .

	What	How
硬體信任根	• 金鑰（儲存） • 熵源	• 用於金鑰存儲的 NVM（OTP） • 靜態熵 (HUK) • 動態熵（TRNG）
硬體信任根(廣義)	• 可信任操作 • 可信任環境	• 防篡改 Anti-Tampering • 安全飛地 Secure Enclave • 金鑰管理 Key Management • 密碼學 Cryptography • 可信固件 Trusted FW/API • 可信作業系統 Trusted OS

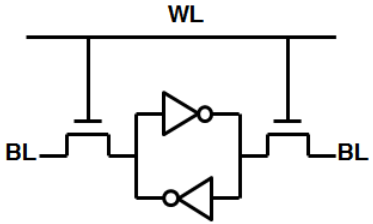
Entropy: Static Entropy a.k.a. PUF .

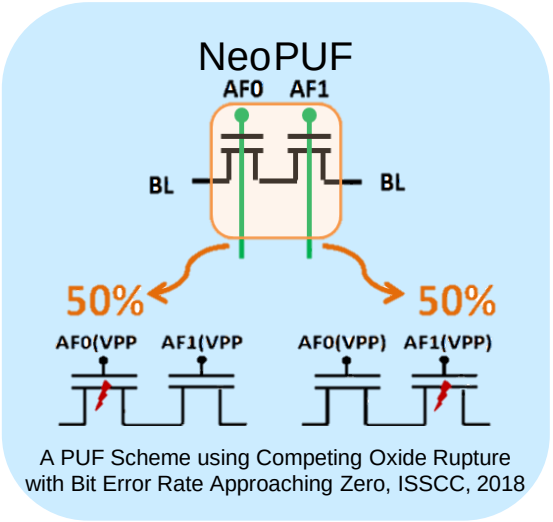
Paired Device with one-time uncertain mechanism
Ex: SRAM-PUF, RO-PUF NeoPUF

Static Entropy

Post-Processing /Conditioning

Static Random Numbers (min.Entropy=1)





A PUF Scheme using Competing Oxide Rupture with Bit Error Rate Approaching Zero, ISSCC, 2018

Entropy Accumulation

HW Adjustment

Health Check

Root Key/HUK

UID

Entropy: Dynamic Source to TRNG .

Device with time-dependence uncertain mechanism

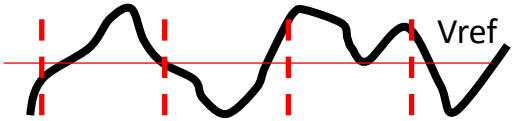
Dynamic Entropy

Post-Processing /Conditioning

Non-deterministic Random Numbers (min.Entropy=1)

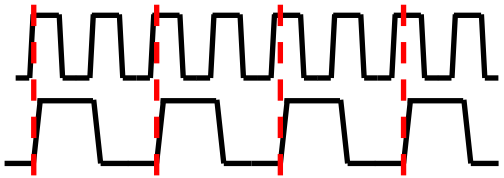
Ex: ROSC, PLL, noises…

Thermal Noises



0, 0, 1, 1….

ROSC



0, 1, 1, 1….

Entropy Accumulation

HW Adjustment

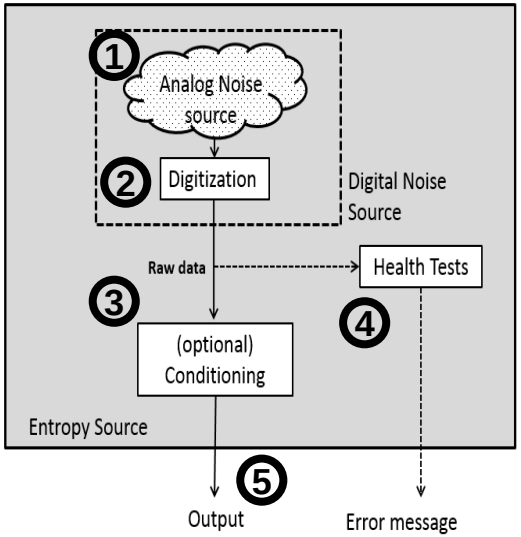
Health Check

Nonce

Session Keys

Entropy: TRNG Design .

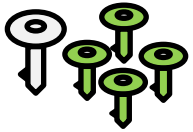
NIST SP800-90B Diagram



90B Requirement	How?
① **Analog Noise Source**	Noise source from PLL/ROSC(Jitter) or ADC
② **Digitization**	Noise to Raw Entropy
①② **Noise Source Analysis**	Entropy Model
③ **Conditioning**	Non-linear Compression (ex: SHA)
④ **Health Test**	Noise source health check
⑤ **Output Data Q.C.**	Passed NIST800-22; 90B Restart Tests Passed AIS31 Tests

The design of either static or dynamic entropy is based on a combination of analog design (noise source) and digital design (function/algorithm).

Key Management: Generation and Access .

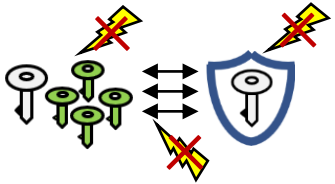


Generation

- UID, TRNG, and Provisioned Key Storage
- KWP, KDF and Key Exchange

Secure Store

- Secure Certified OTP with Tamper-Proof
- Static Entropy Protects OTP



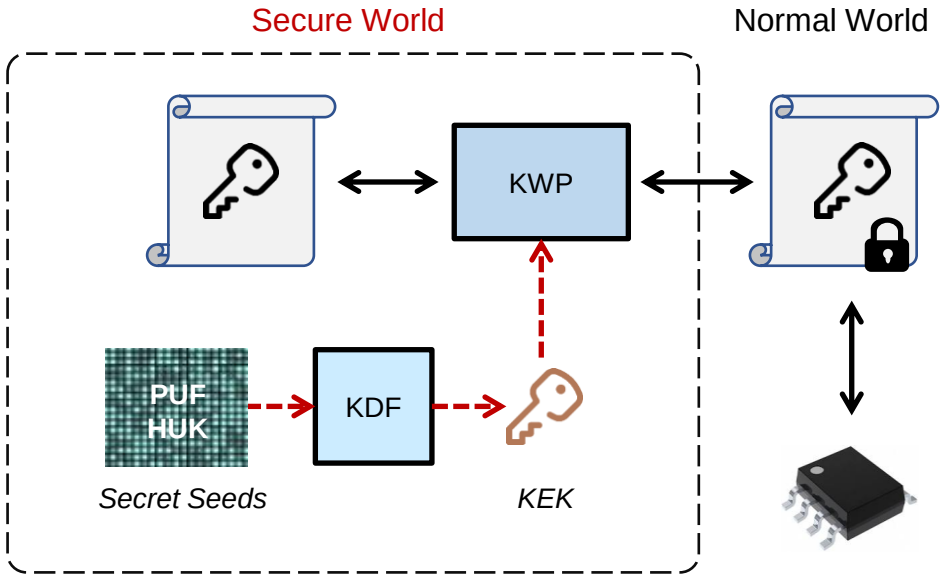
Use

- PUF and HW cryptos Protect Keys
- Access Authority Control for Keys

Revocation

- Complete Key/Data Lock or Revocation
- Entropy Revocation

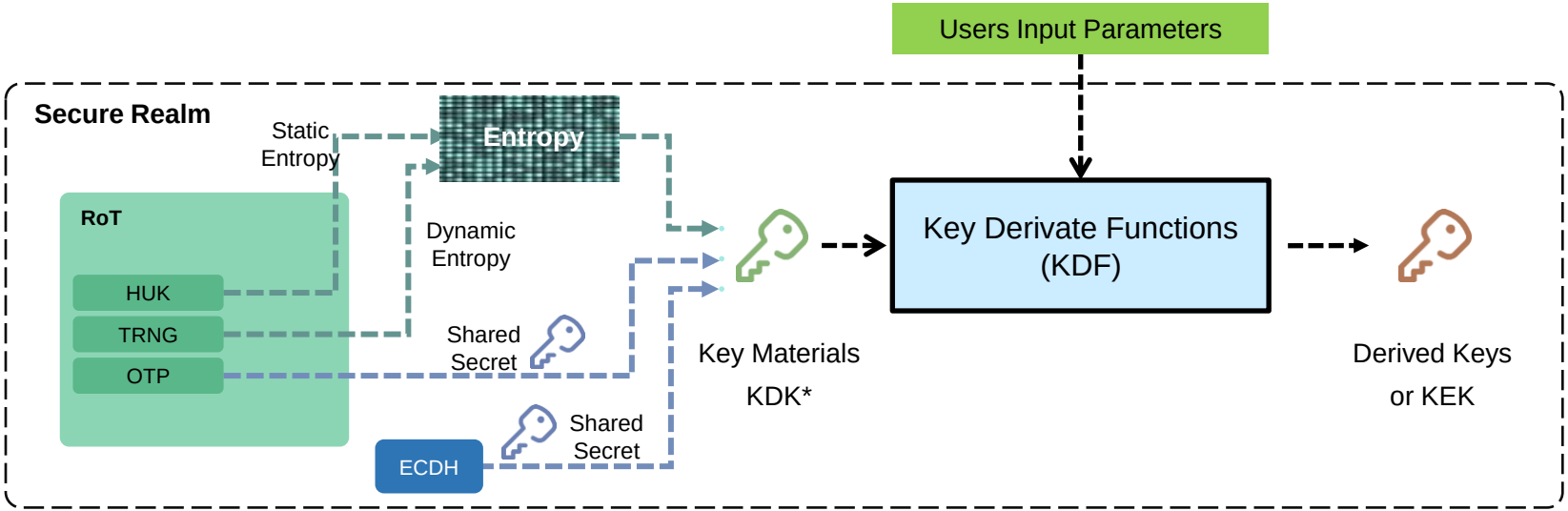
Key Management: Key Wrapping .



- Only encrypted key stored in normal world
- KWP uses KEK from PUF or HUK seed

KWP: Key Wrapping Function; KEK: Key Encryption Key; KDF: Key Derivation Function

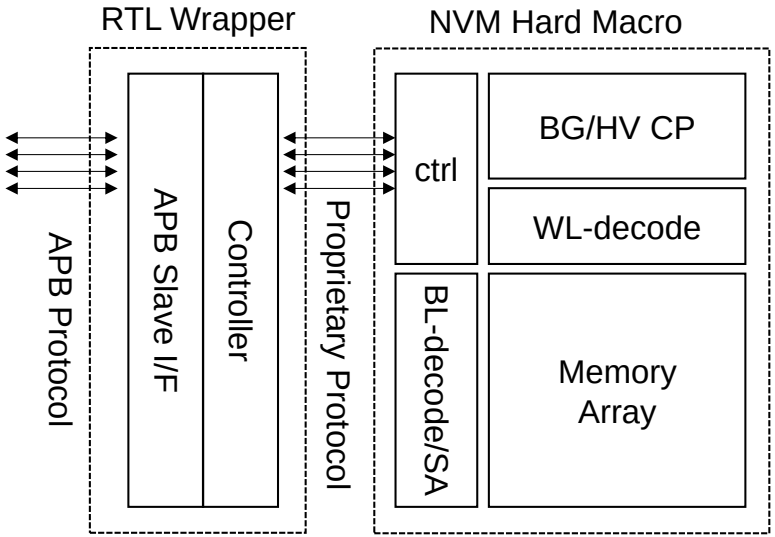
Key Management: Key Derivation Function .



- KDF in PUFiot follows standard NIST SP800-56C
- Derived keys are stored in MMU (memory management unit) for further use or as KEK to wrap the key for storing outside the PUF-realm

KDK: Key Derived Key; KEK: Key Encryption Key

Secure Storage: Conventional Storage .



Invasive/Semi-invasive attacks

- SEM, FIB, TEM on bit-cells
- Passive voltage contrast by SEM on bit-cells
- Locate address, delayer and nano probing

Non-invasive attacks

- Side channel attacks by CPA/DPA
- Unauthorized access
- Fault injection on visible pins
- Fault injection on secure registers
- Photon emission inspection (InGaAs/EMMI)

Secure Storage: Threats and Countermeasures .

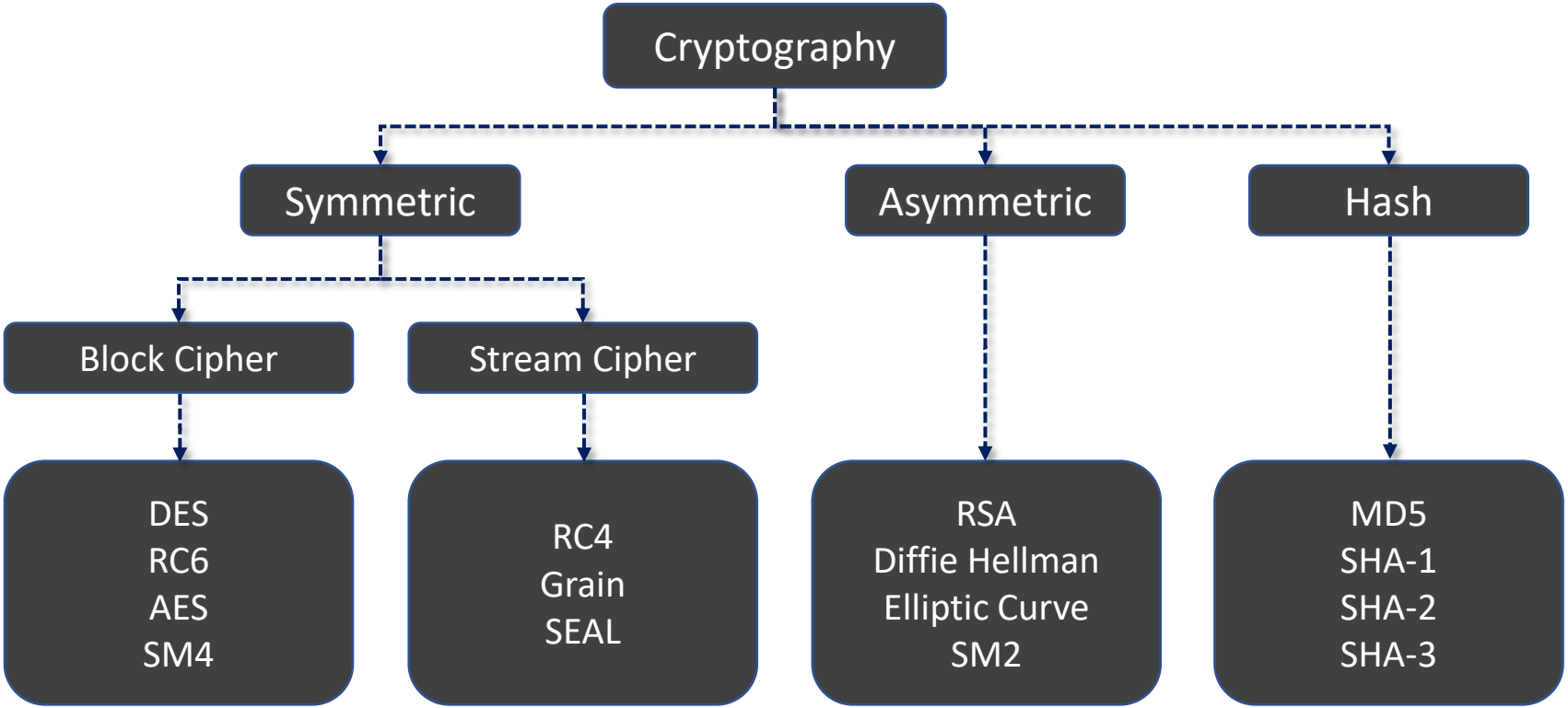
Threat Model	Against Design (Green/Blue for hard macro/digital)
SEM, FIB, TEM, optical inspection (OBIC/OBIRCH)	Need to apply secure device design
Passive voltage contrast	Need to apply special layout
Locate address, delayer and **nano probing optical inspection (OBIC/OBIRCH)**	Top metal shielding, security-oriented IP layout, inter-metal routing
	Encryption, post-masking, random dummy read
Power analysis on SA during read	Active SA protection during reading
Fault injection on IO or mode select	Output Data Fault Detection, Pin Protection
Rollback, **replay attack** and **software access**	Assess Permission Design and Post-Masking
Secure setting or reserved bit **leakage**/revise	Secure repair and test-mode protection by lock
Key location, photon emission, fault injection, glitch	Random dummy insertion READ
Power analysis on CP or maliciously **cut power**	Unified operating power and power floating detection
Photon emission inspection	Active SA protection during reading
	Address/IO scrambler, post-masking, random dummy read

硬體安全設計與操作

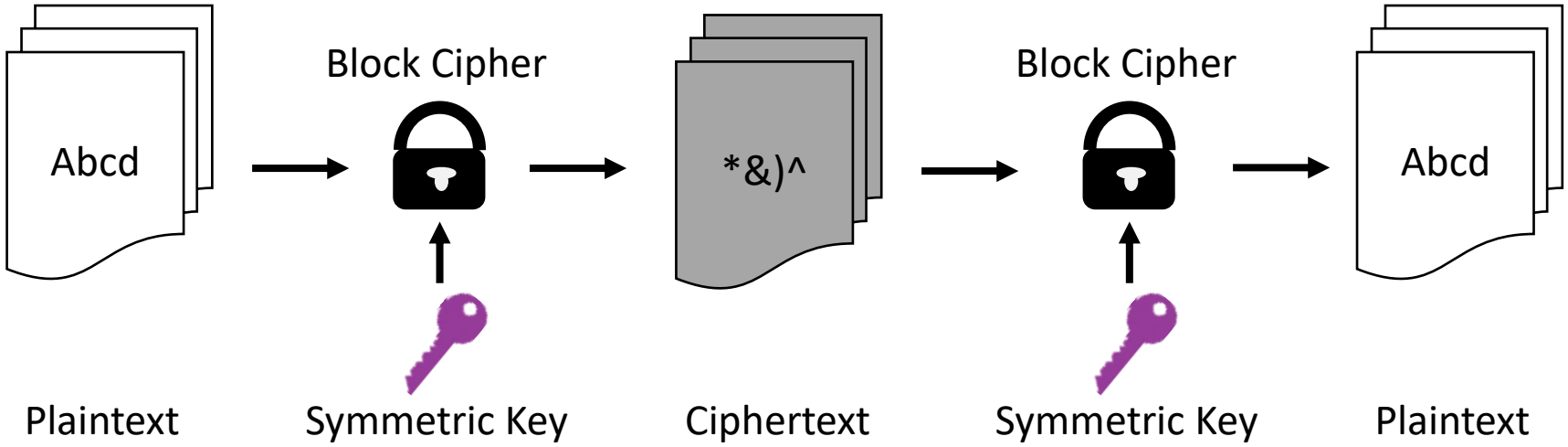
Symmetric and Hashing Cryptos for Secure Operations

→ Block Cipher, Digest and Authentication

Cryptographic Algorithms .



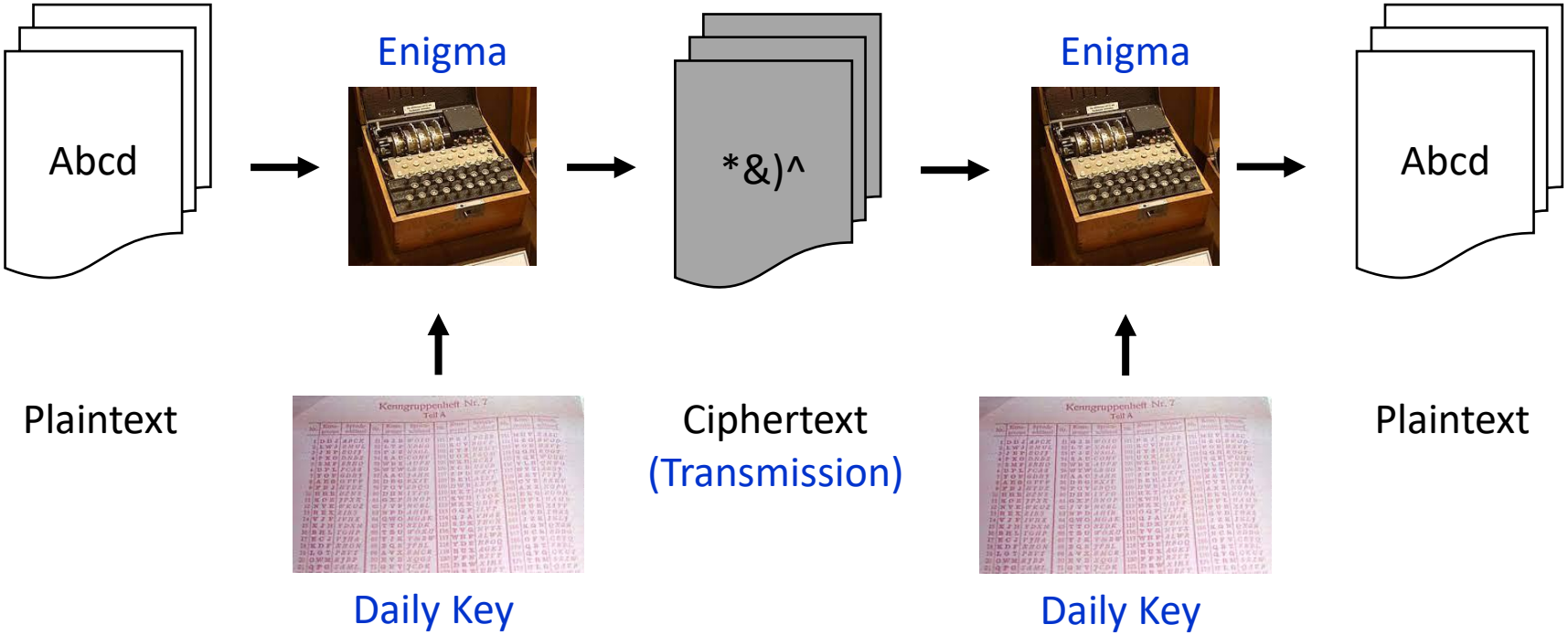
Symmetric Cryptography .



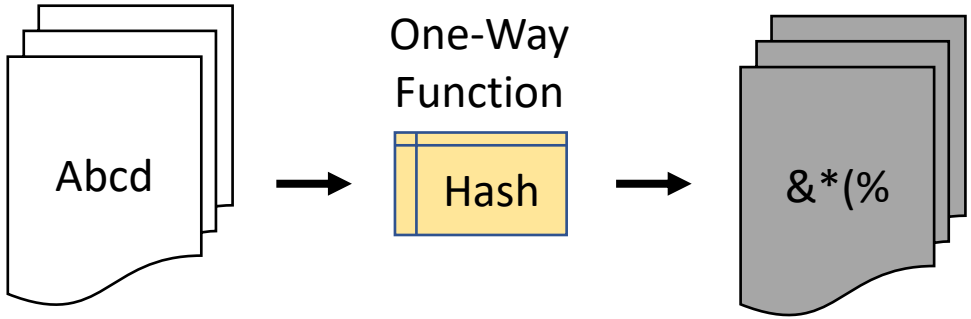
Algorithms : AES、ChaCha20、3DES、DES、RC5、RC6、SM4

Use Cases : Secure data storage. Real-time data encryption/decryption

Example for using Symmetric Cryptography .



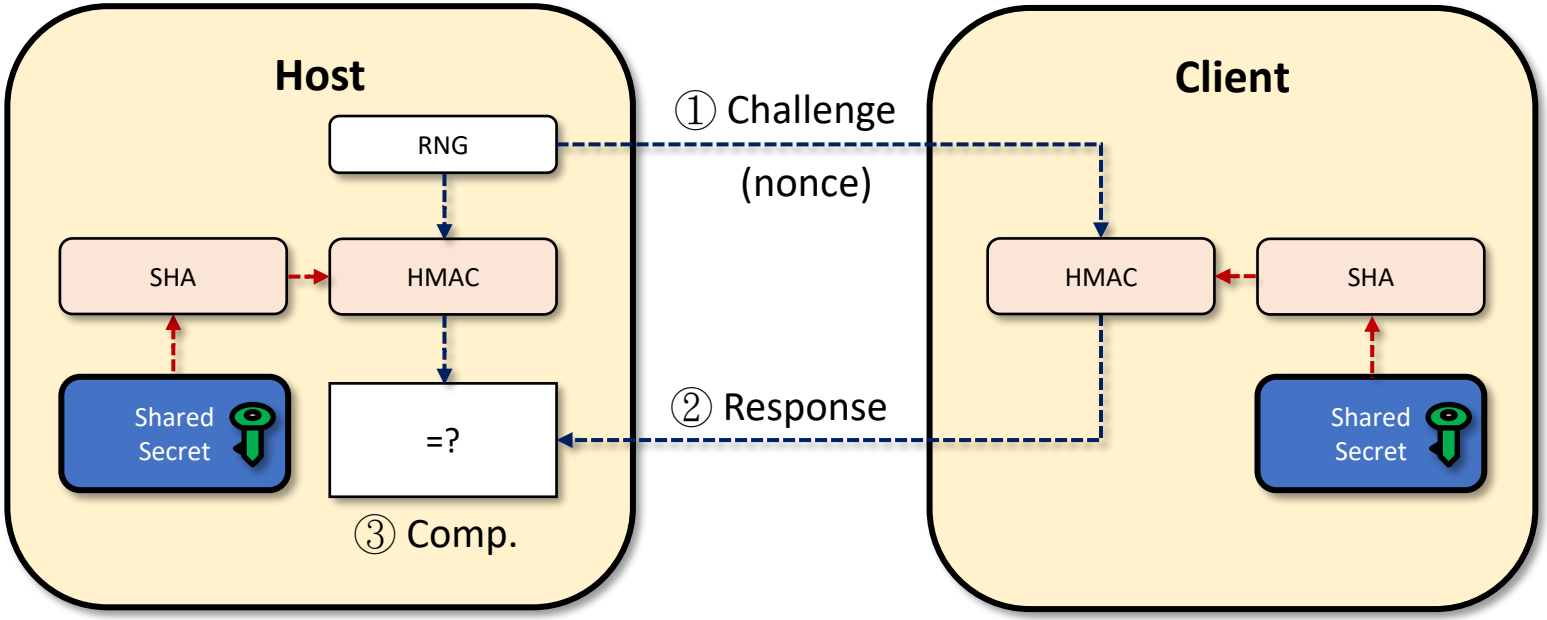
Hash Function .



Algorithms : MD5、SHA-1、 SHA-2、 SHA-3、SM3

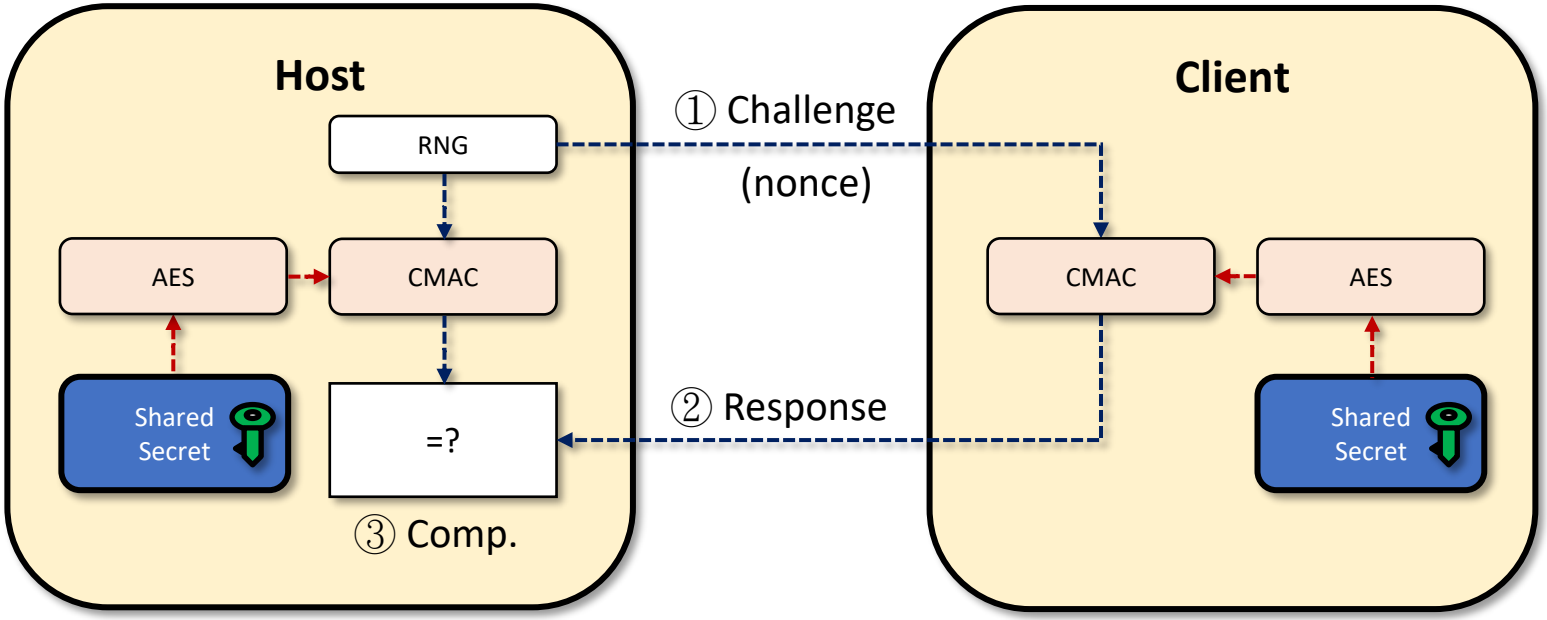
Use Cases : Check data integrity (ex. downloading)

Symmetric Authentication (HMAC) .



- Nonce is required to prevent replay attack.
- The same procedure is needed if the client needs to verify the shared key of Host

Symmetric Authentication (CMAC) .



- Nonce is required to prevent replay attack.
- The same procedure is needed if the client needs to verify the shared key of Host

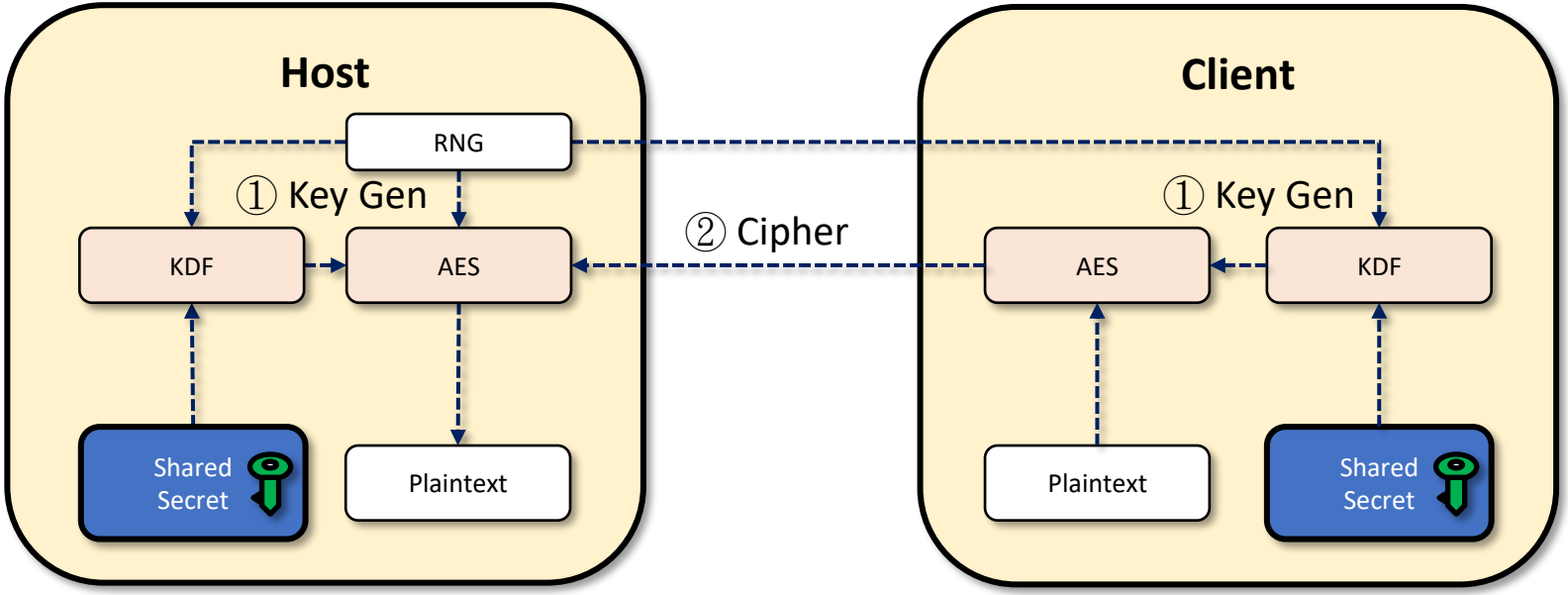
Wrap-up: Symmetric Authentication .

Pros: Easy and Fast Authentication

Consideration:

1. How to inject secret key in Host and Client?
2. How to protect secret key in Host and Client?
3. Only suitable for one-to-one bundles, not for one-to-many use scenarios.
   (Host needs to record too much information)

Symmetric Secure Communication .



① The host provides nonce for host/client to generate session share key

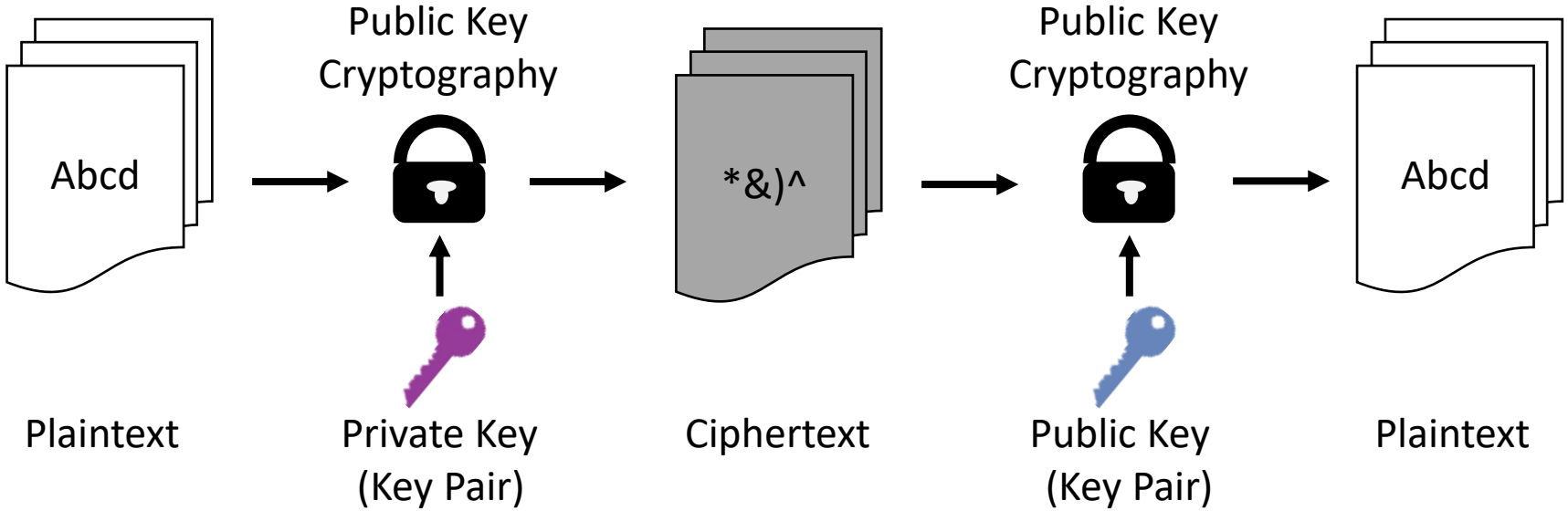
② Client uses AES with temporary share key to encrypt the plaintext into cipher

硬體安全設計與操作

Asymmetric Cryptos for Secure Operations

→ PKC, Authentication and Certificate

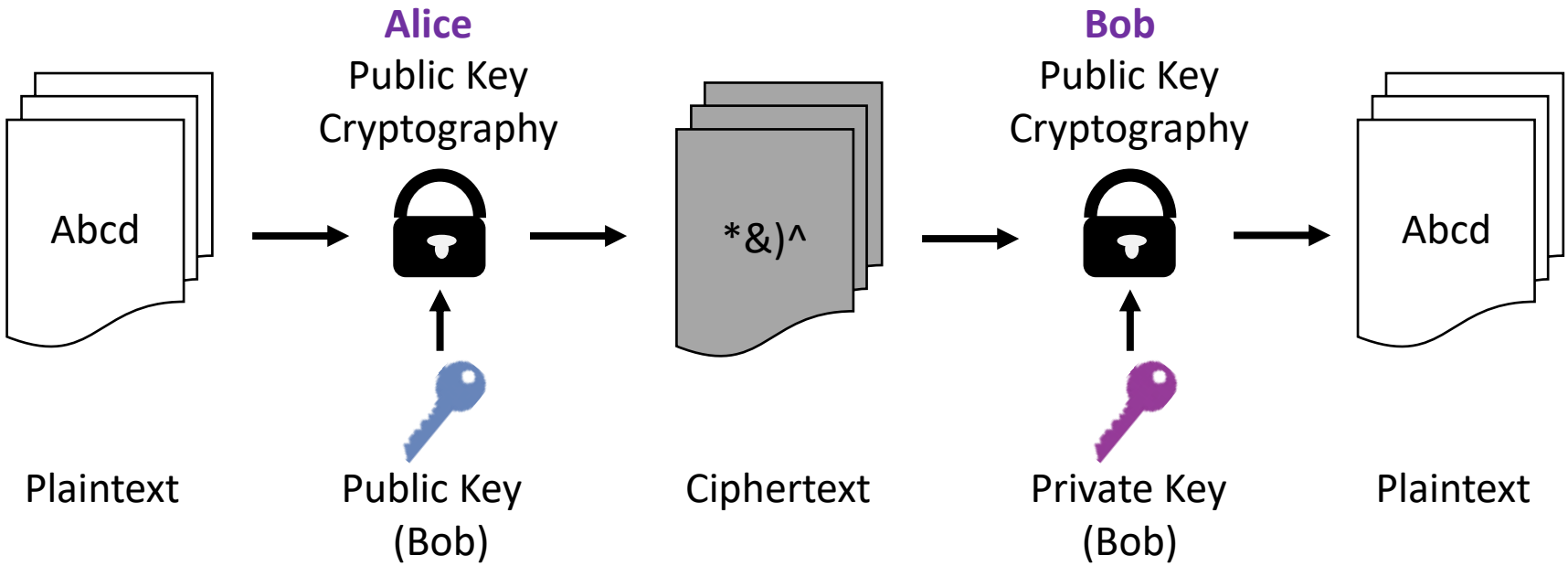
Asymmetric Cryptography .



Algorithms: RSA / ECC / SM2

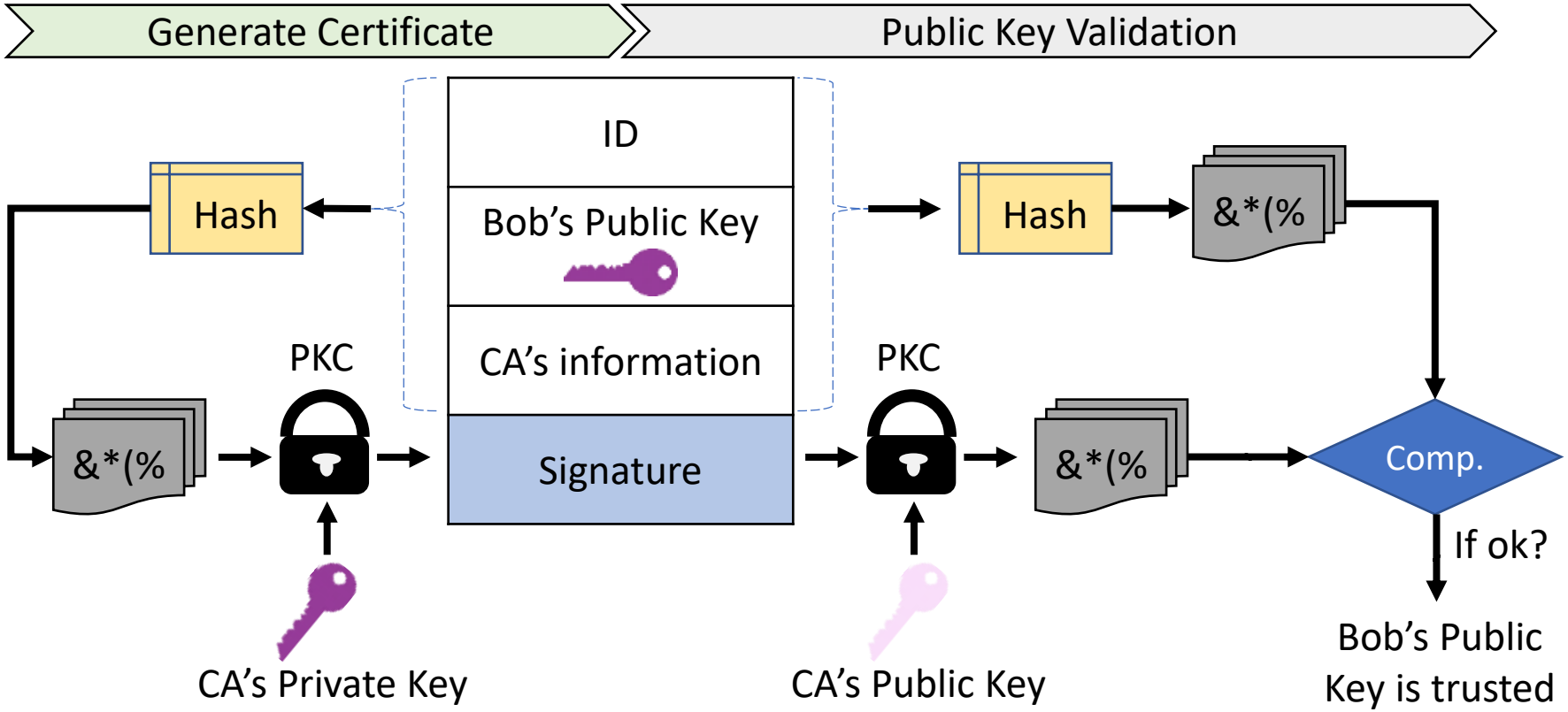
Use Cases: Data authentication. Secure email exchange (PGP email encryption)

Public Key Concerns .

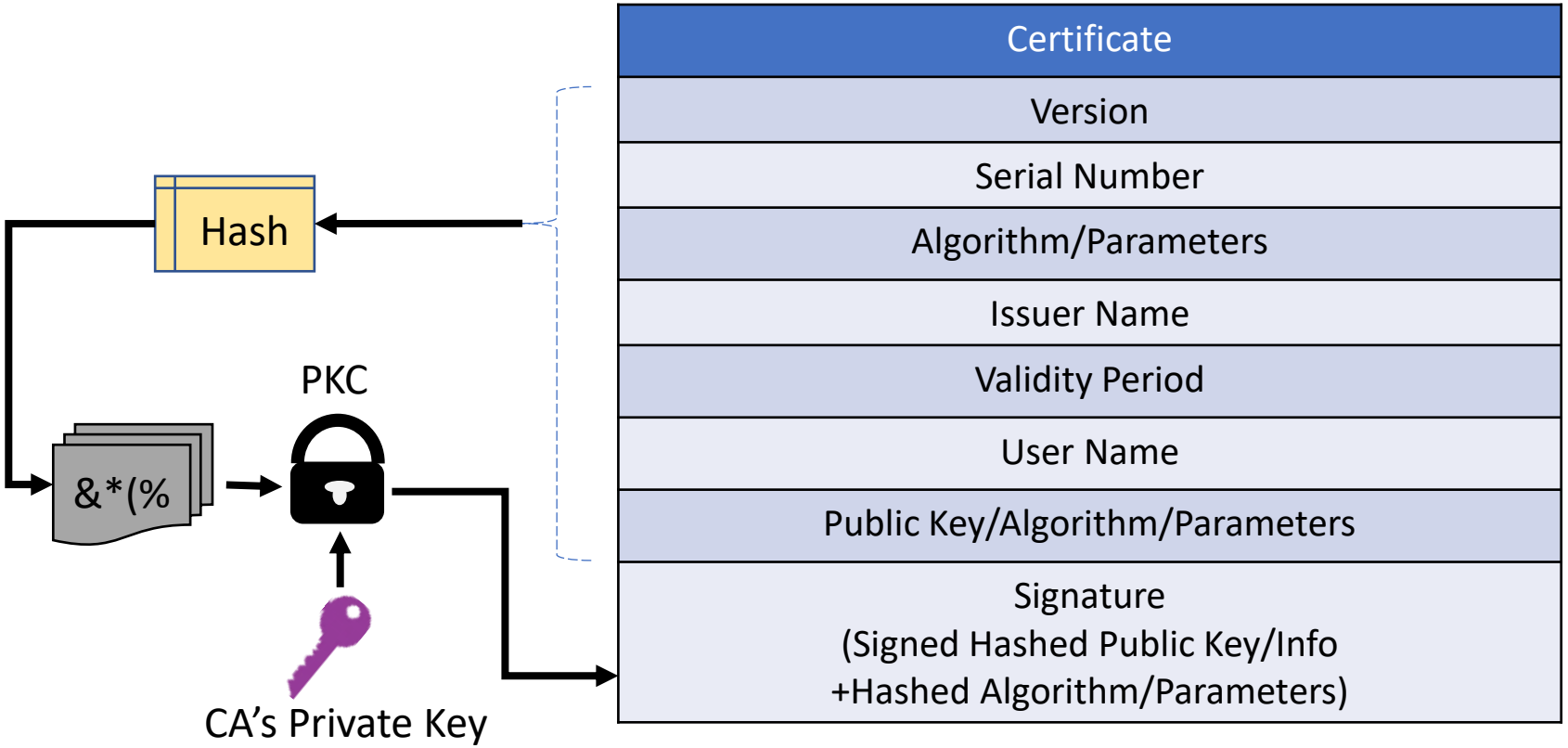


Alice wants to use Bob's public key to send data to Bob, but what if Bob's public key is fake? → Bob's public key need to be **authenticated** by 3rd party (CA)

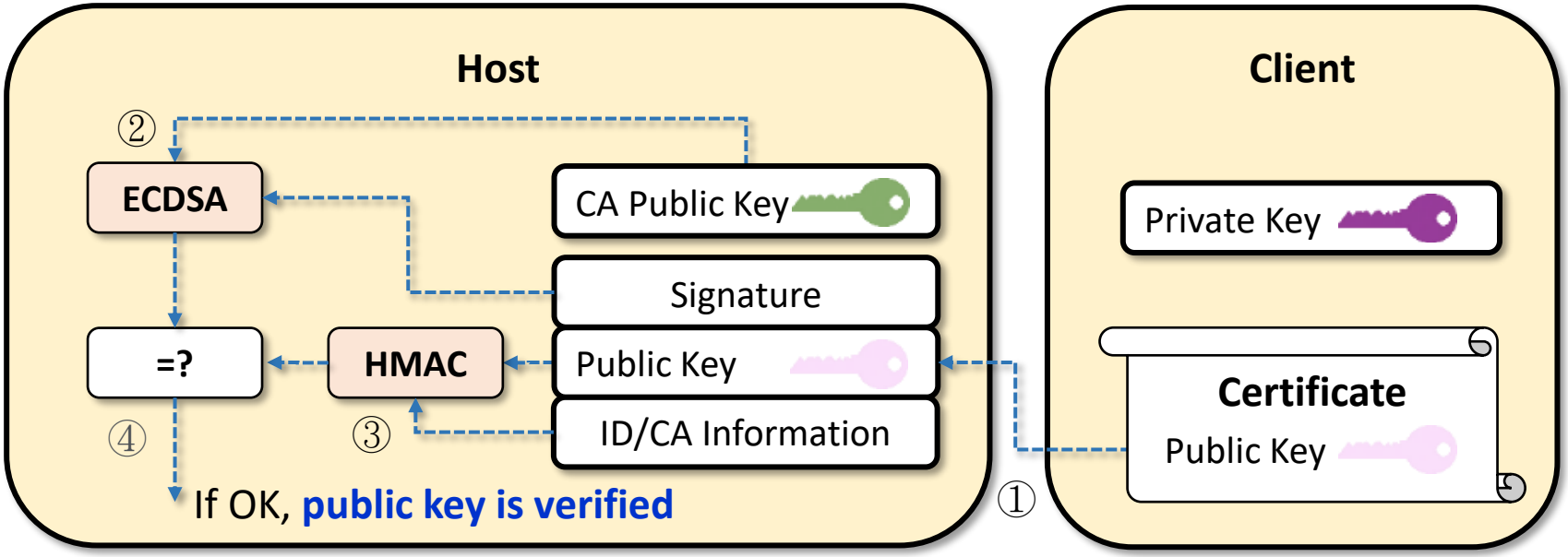
Certificate with Signed Public Key .



Certificate with X.509 Format .

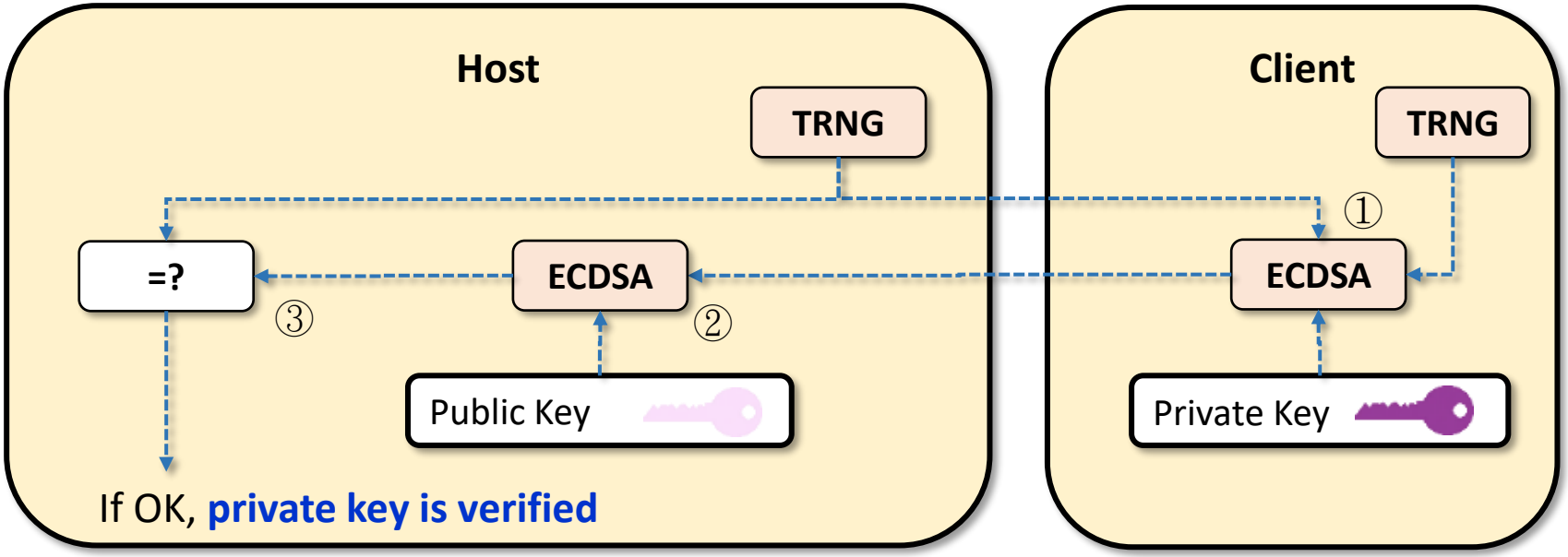


Asymmetric Authentication – Phase I .



① The host requests a certificate (ID/public key/signature) from the client.
② Host uses ECDSA on the signature with the CA public key and then gets the digest
③ Host uses HMAC to get another digest of the ID/public key/message
④ Compare the two digests to verify the validity of the public key

Asymmetric Authentication – Phase II .



① The client uses its private key to sign the nonce from the host's TRNG
② The host uses the client's public key to decrypt the signed nonce from the client
③ Compare the initial nonce with the decrypted nonce, if OK, the private key is verified.

Asymmetric Authentication : MFP and Cartridge .

https://news.tvbs.com.tw/life/328610

Wrap-up: Asymmetric Authentication .

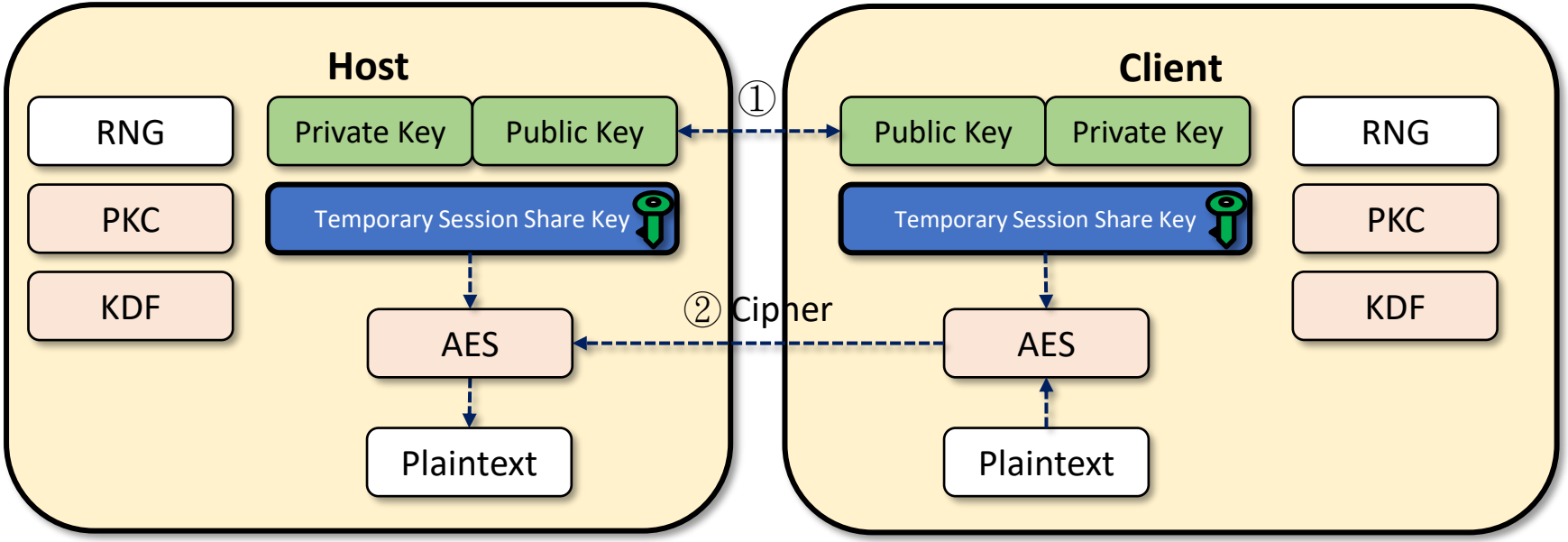
Pros:

1. Complete Authentication with proved certificate
2. Fulfill one-to-one bundle and one-to-many bundle scenarios.

Consideration:

1. How to protect private pair in Client? Secure room and HRoT is needed for private key protection
2. RSA or ECC takes time for authentication

Asymmetric Secure Communication .



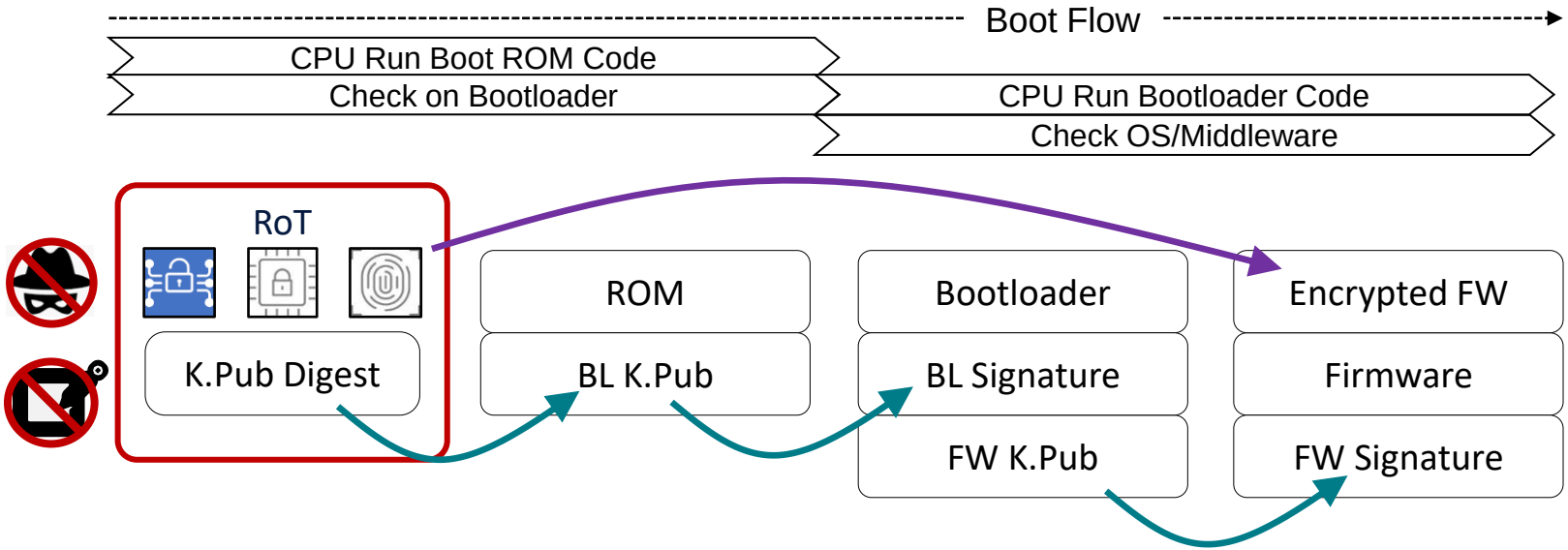
① Public key exchange for share key generation using ECDH algorithm

② Client uses AES with session ECDH key to encrypt the plaintext into cipher

硬體安全設計與操作

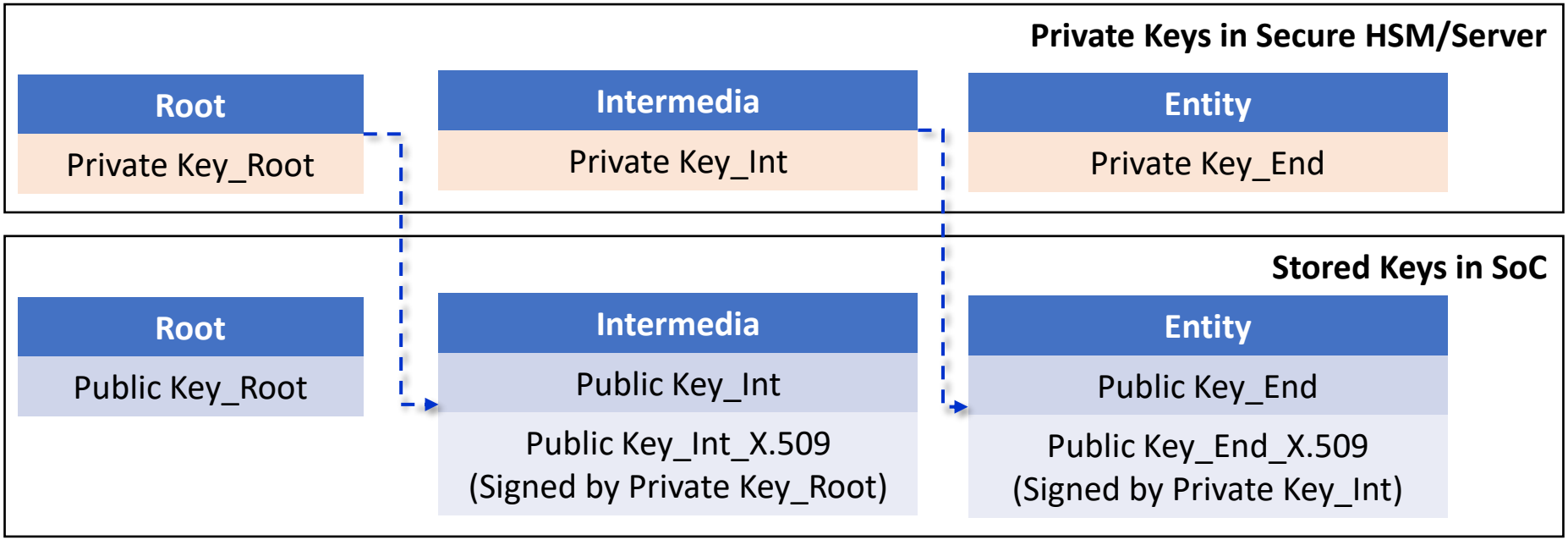
Secure Boot

Internal Security: Secure Boot .

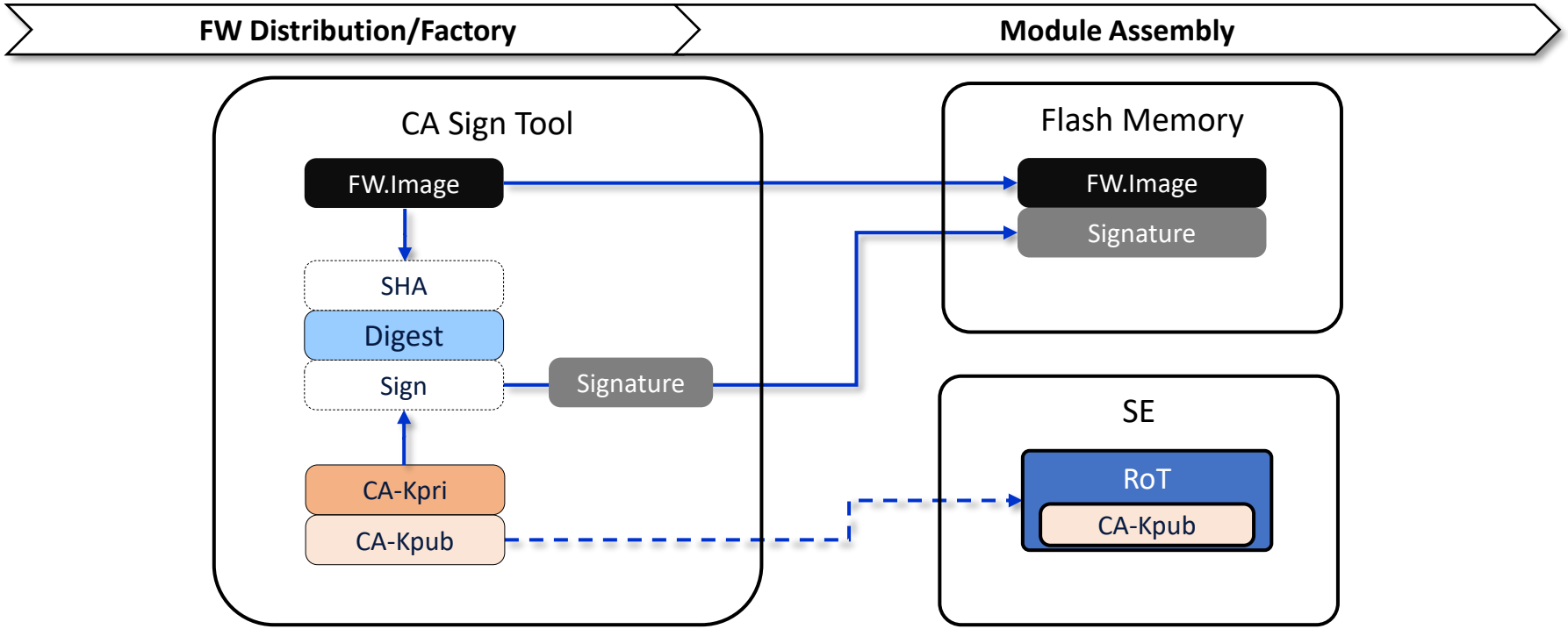


- RSA/ECC and SHA for FW digest and verification → Anti-FW hacking
- FW protection using AES with local PUF key → Anti-counterfeiting

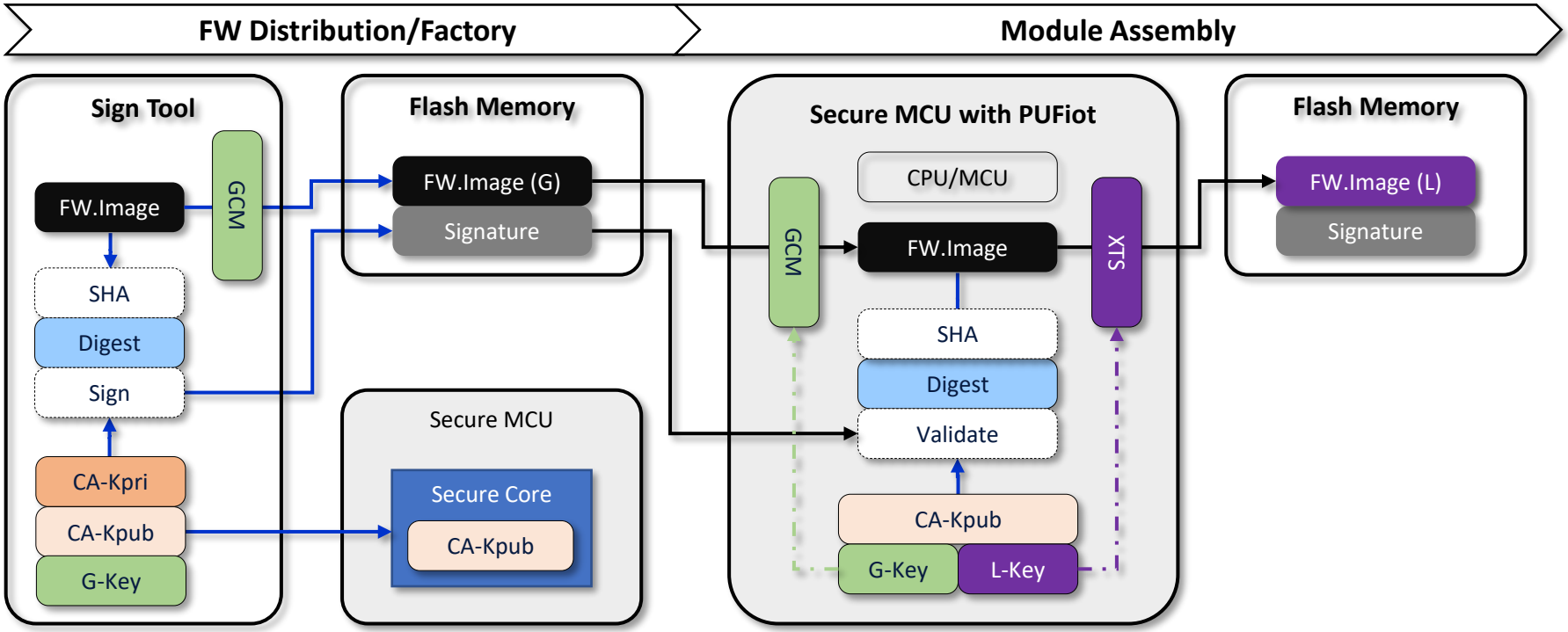
Chain of Trust in a SoC (Key Generation) .



Common Plain FW Deployment Flow .

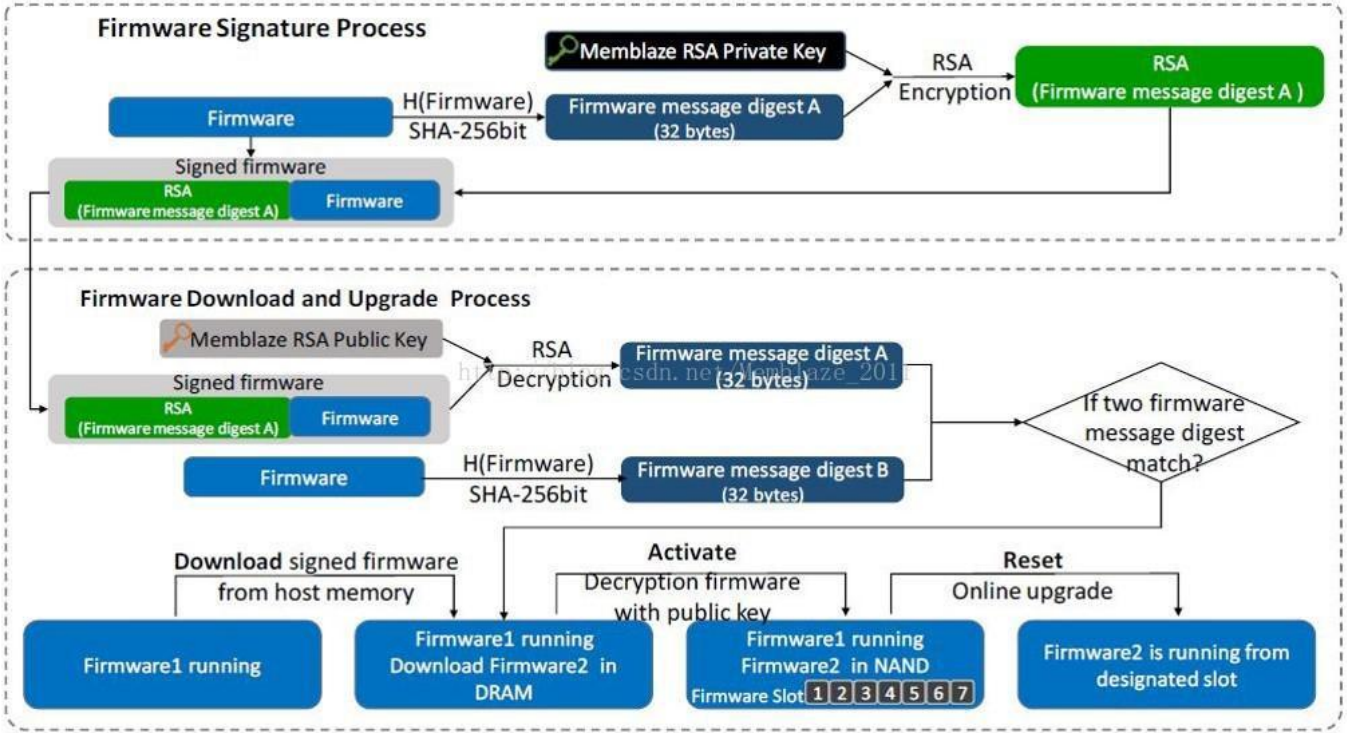


Device Pairing Secure FW Deployment .



- **Secure FW deployment using global key management**
- **Paired FW and SoC using local key during assembly and booting-up**

Reference .



https://www.twblogs.net/a/5b7d9a052b71773f4f1812fd