SAND2012-4953C

SECURING TRUSTED EXECUTION ENVIRONMENTS WITH PUF GENERATED SECRET KEYS

2012 International Symposium on Advances in Trusted and Secure Information Systems

Matthew Areno
University of New Mexico
mareno@ece.unm.edu

Jim Plusquellic
University of New Mexico
jimp@ece.unm.edu

Questions being addressed

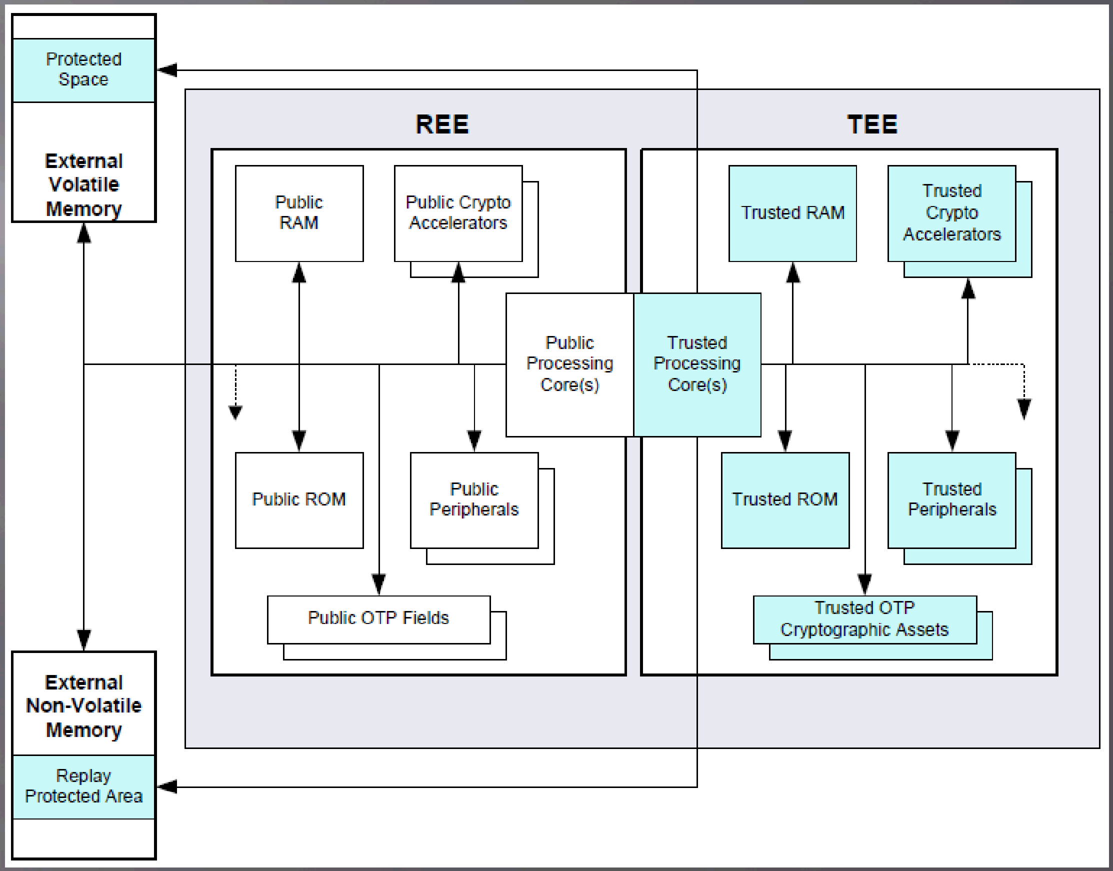
- What are Trusted Execution Environments (TEEs)?
- How do they work with the secure boot process?
- How can a PUF be used to protect a TEE?
- What are the benefits provided with a PUF generated secret key?
- What are the possible applications of this method?
- How will this be proven?

Trusted Execution Environment

- A separate execution environment that is isolated from the Rich-OS execution environment, or REE.
- Currently being standardized by Global Platform.
- Consist of both hardware and software elements, with the hardware providing the assurance of isolation and the software providing communication and execution mechanisms between the two.

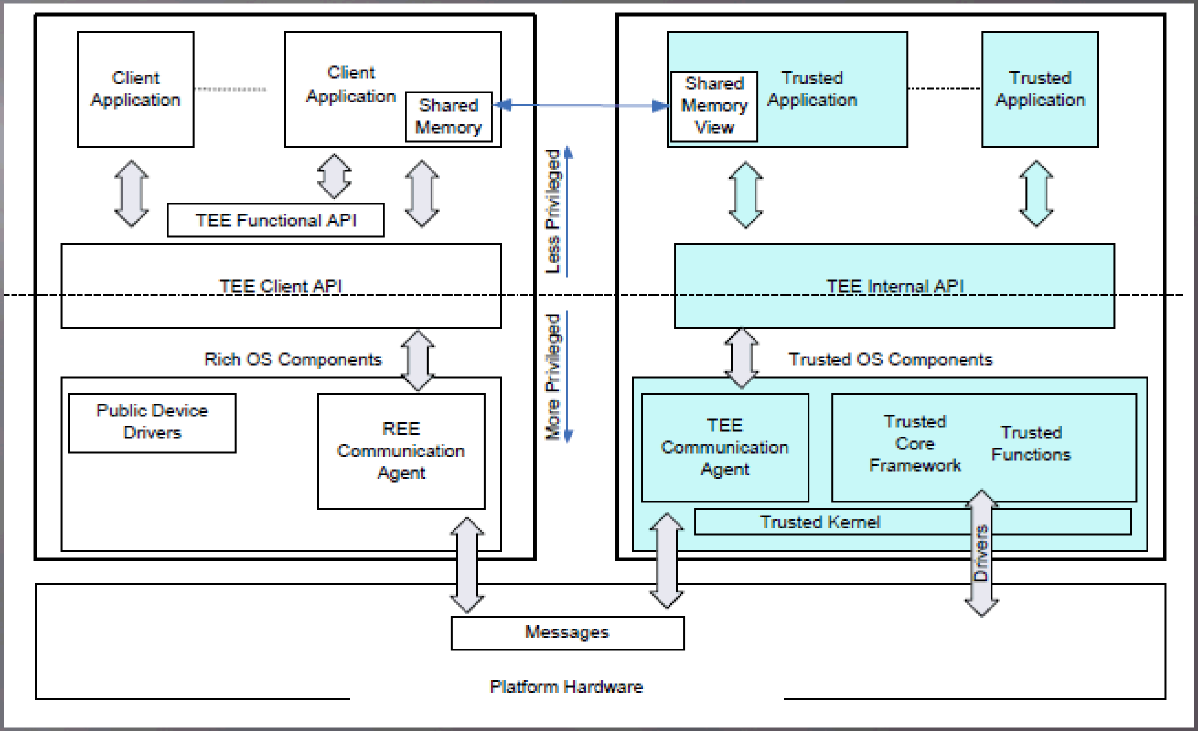
TEE Hardware

- Hardware is partitioned into two levels of functionality: trusted (secure) and public (non-secure)
- Hardware support, via mechanisms such as ARM TrustZone, prohibit access of secure elements by non-secure applications.
- Includes support for separate interrupts, memory partitioning, and world switches.

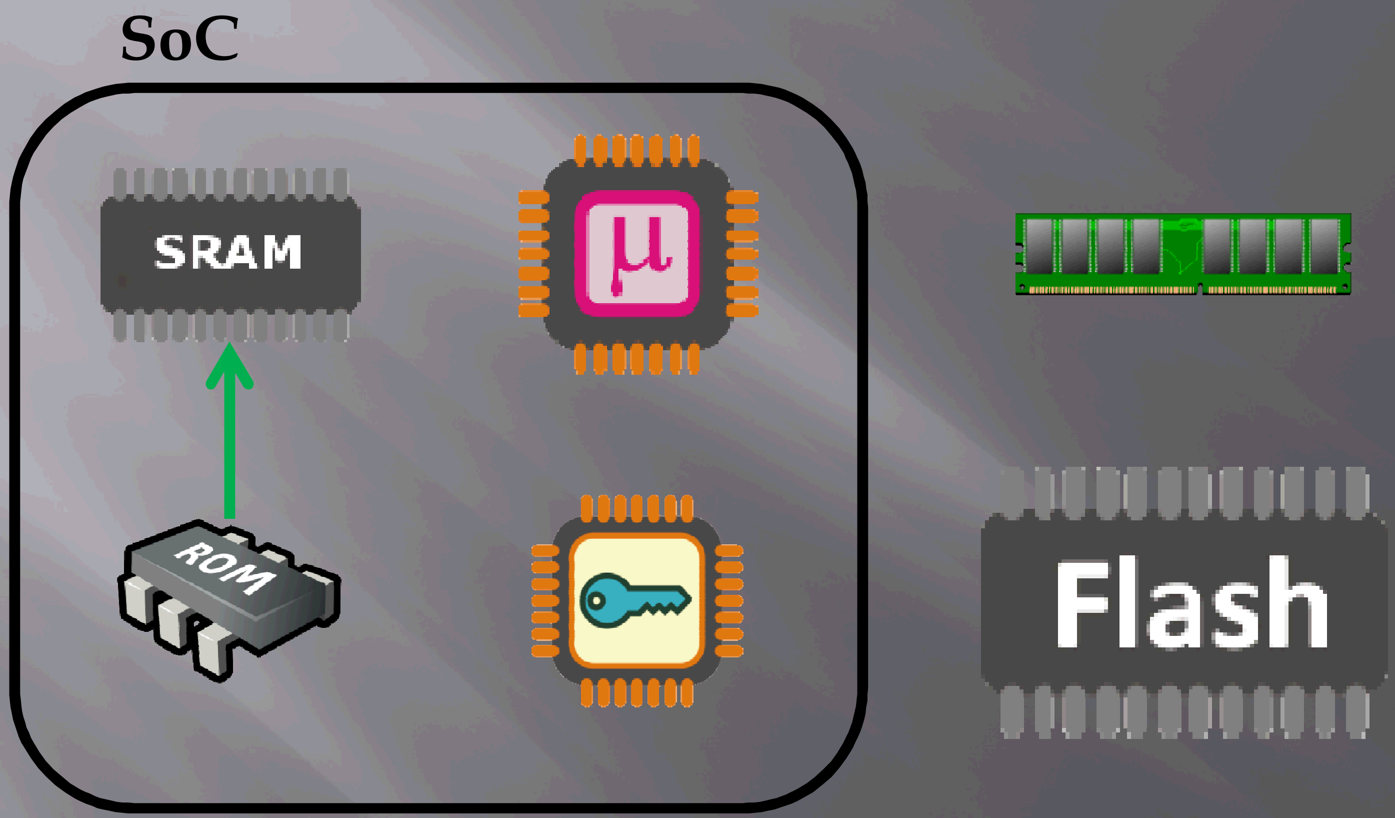


TEE Software

- Software API provides access to trusted (secure) element from the REE.
- World switches handled via monitor code invoked through SMC instruction.
- Additional communication and data sharing available through TEE/REE shared memory channel.

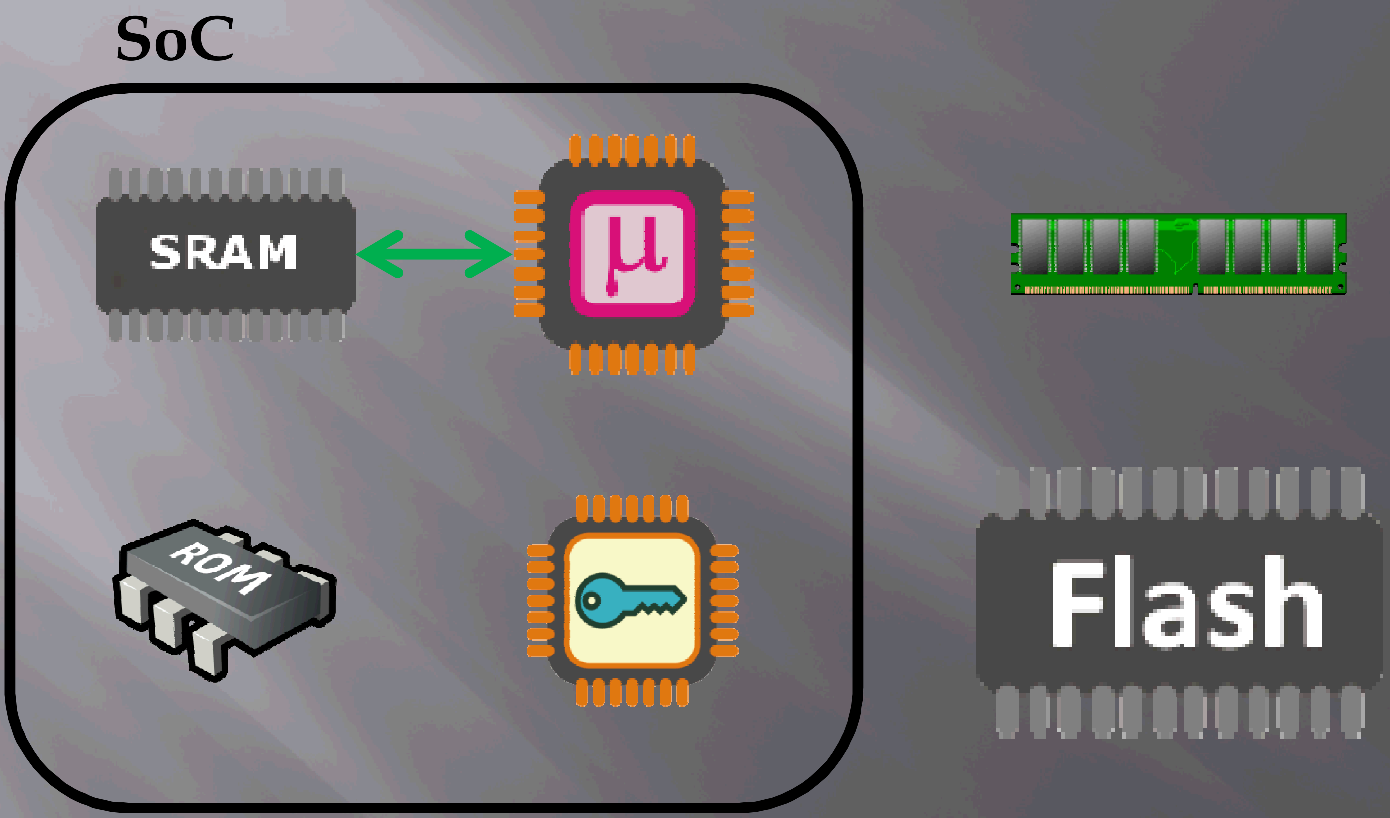


Typical Process



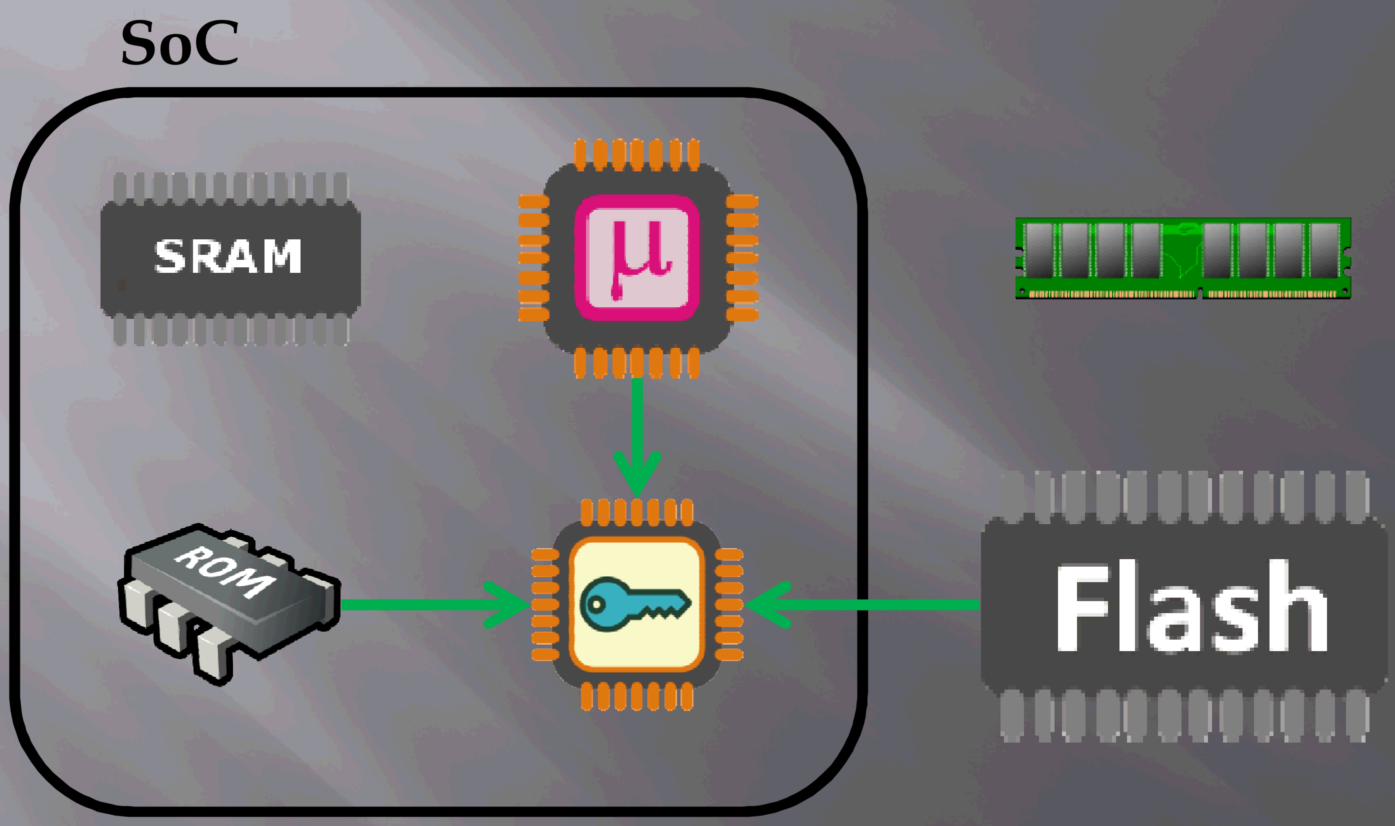
The first step is pulling the boot-rom from some form of read-only memory and loading it into secure RAM.

Typical Process



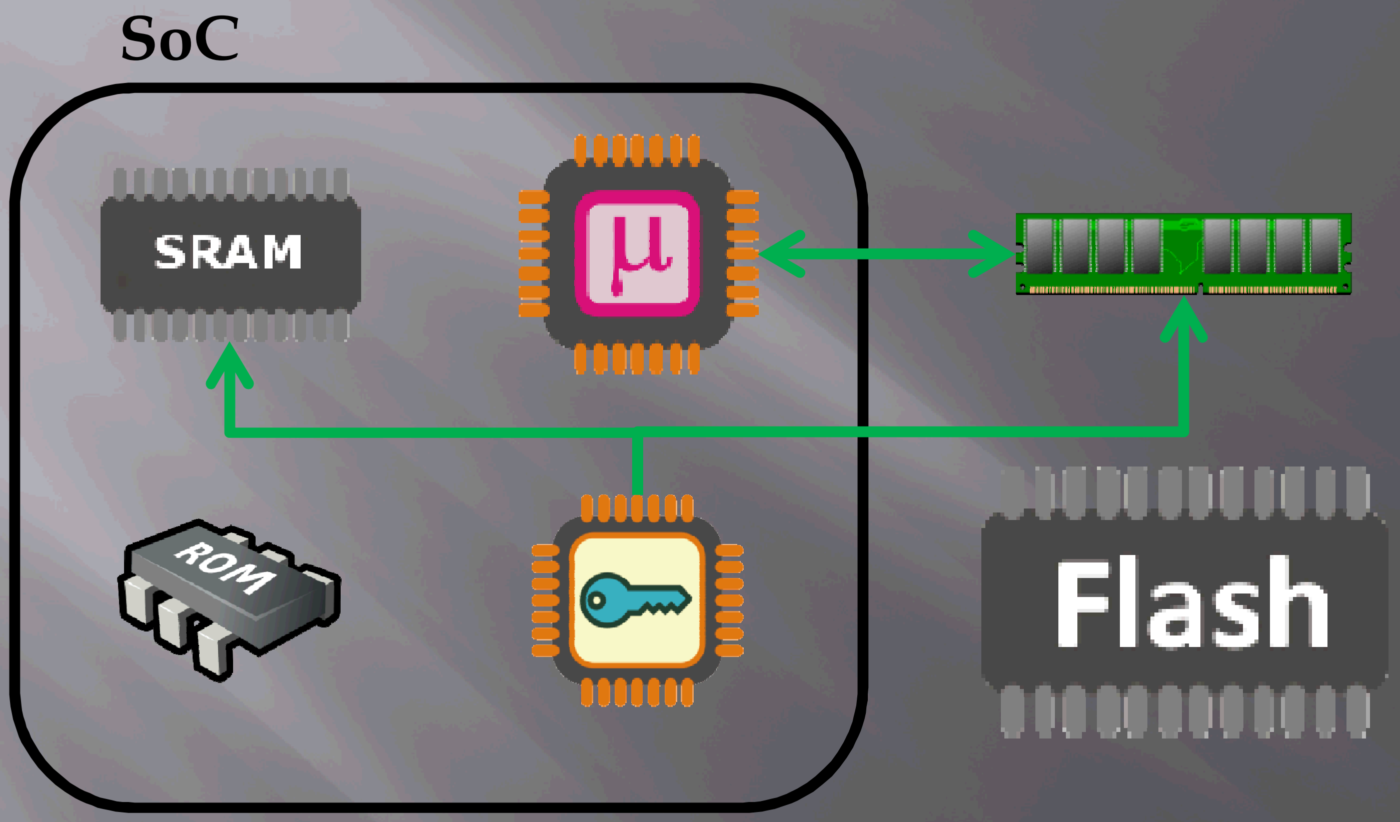
Next, the microprocessor begins execution out of the secure RAM.

Typical Process



The microprocessor then sends a command to the crypto engine to retrieve a secret key from some form of ROM and decrypt the 1st stage bootloader from flash.

Typical Process



The resulting decrypted bootloader is then loaded into either SRAM or standard DRAM and execution continues.

Secure Boot Process

- For secure boot, the previous process is used with the addition of two extra elements: measurement and verification.
- After decrypting each stage, the code is measured, typically with some type of hash algorithm, such as SHA-1.
- The result is then compared with a known correct value (typically stored in NVROM).
- If they match, bootup continues. Otherwise, either the boot is canceled or it continues in a non-secure state.

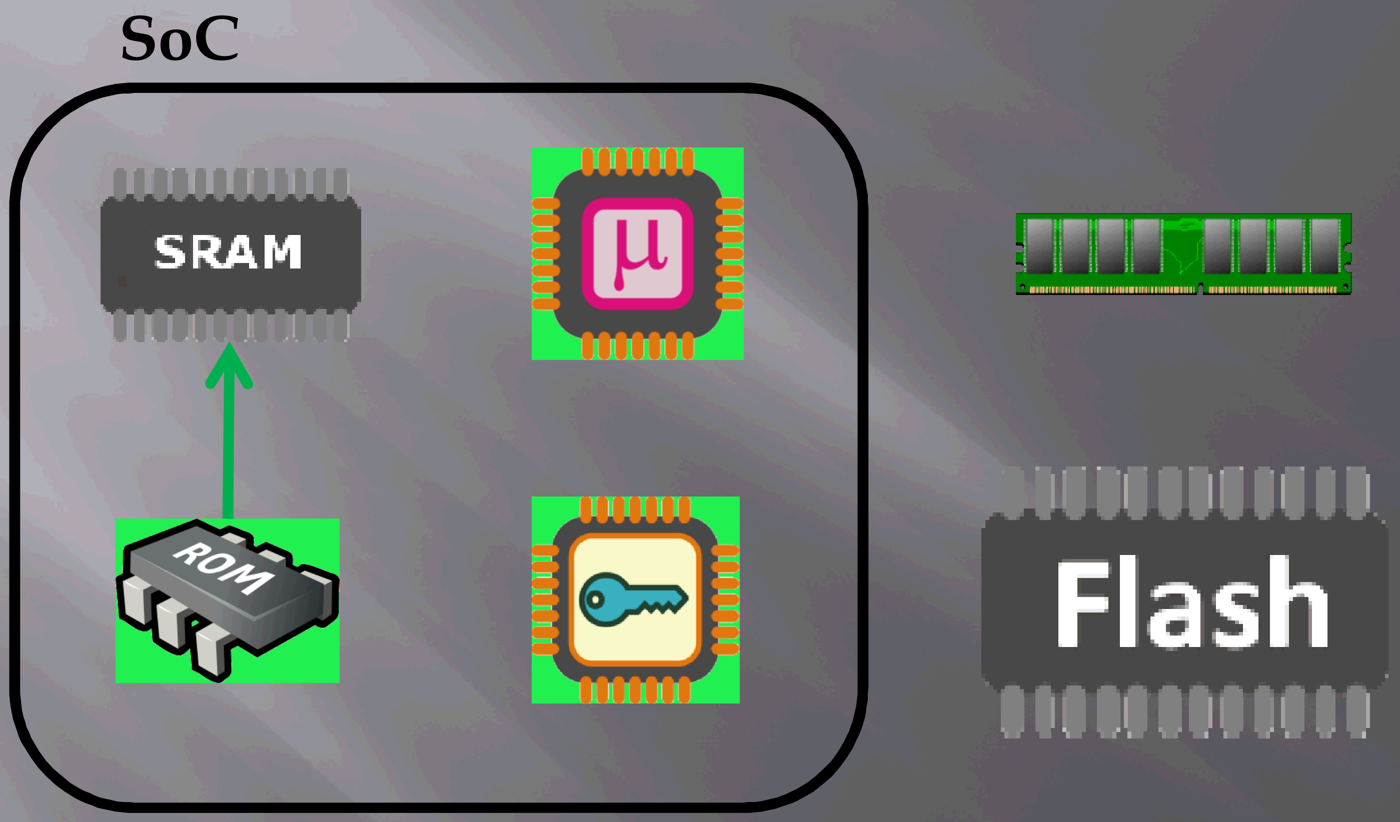
Secure Boot with TEEs

- In a secure boot environment, TEEs are treated similarly to bootloaders.
- Because the measurement value for the TEE must be stored in NVROM, this prohibits alteration or modification of the TEE.
- Further, TEEs will typically be identical for multiple devices.
- If the TEE is encrypted on the device, that encryption key will also be identical for multiple devices.

Securing TEEs with PUF Keys

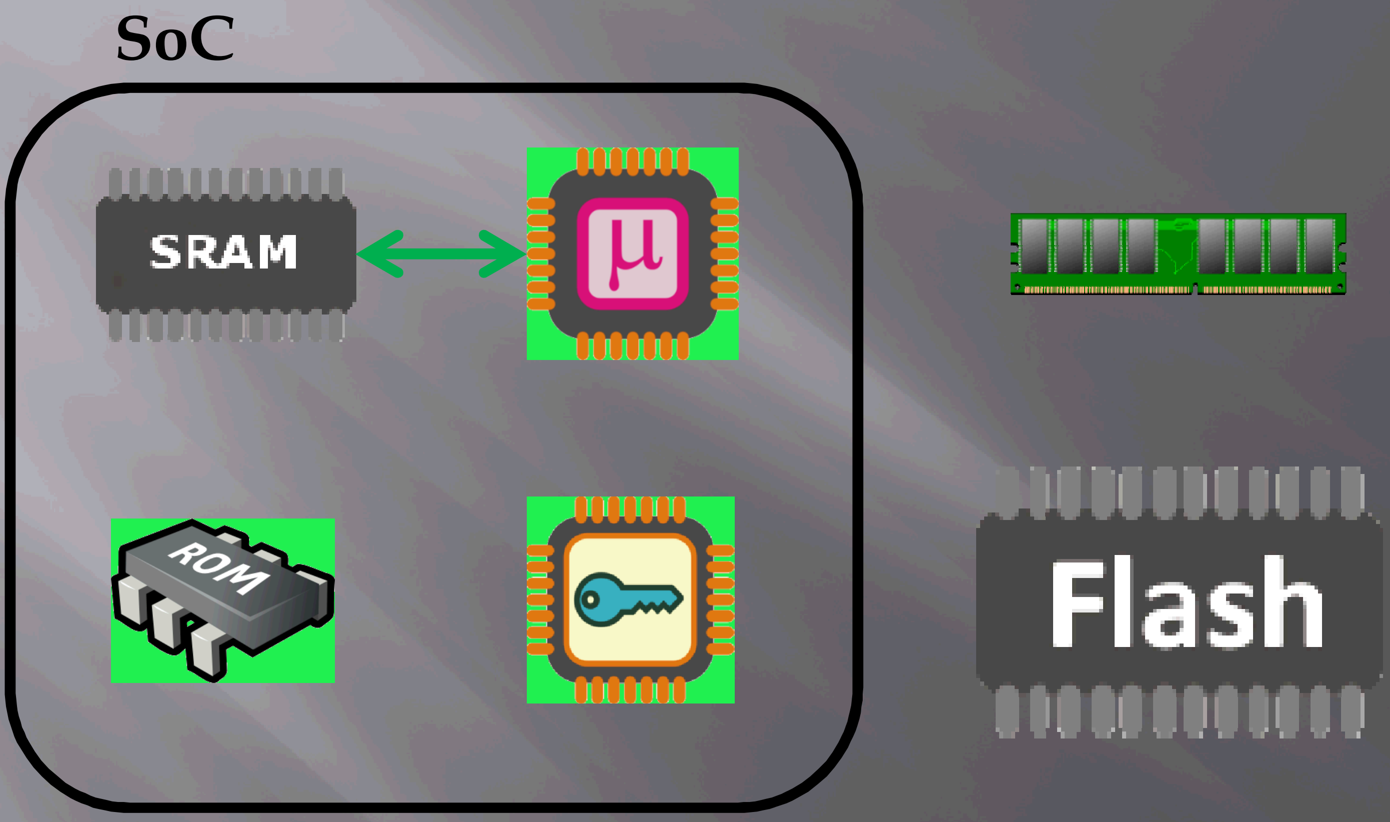
- A PUF, or Physically Unclonable Function, is capable of generating unique-per-device security keys.
- By incorporating a PUF generated key into the SoC cryptographic engine, we can provide the ability to encrypt/decrypt the TEE for each device in a unique manner.
- Further, the PUF generated key can be used to encrypt the measurement value of the TEE, allowing the TEE to be modified.

Secure Boot Architecture with PUF



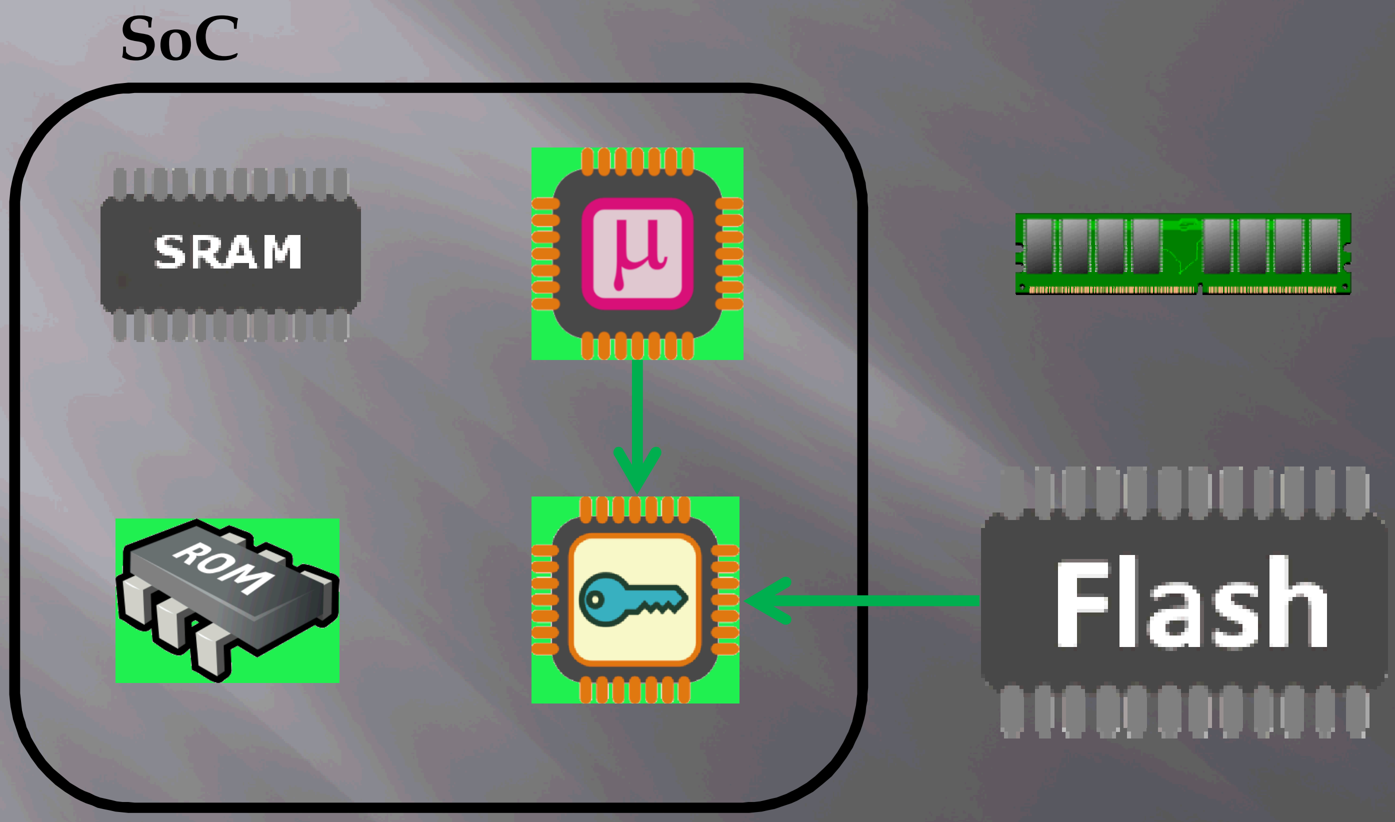
Devices supporting PUF generated keys.

Secure Boot Architecture with PUF



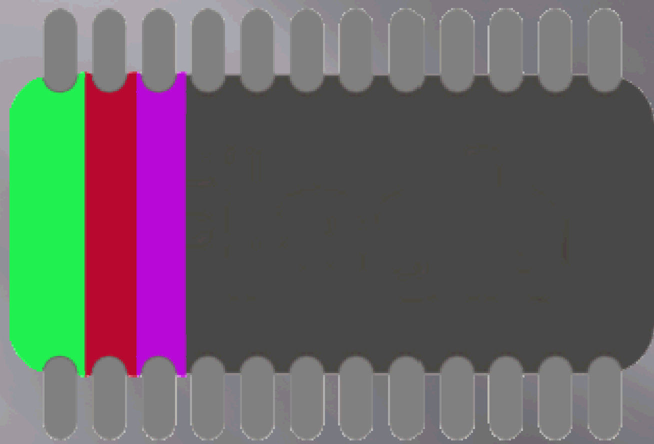
Devices supporting PUF generated keys.

Secure Boot Architecture with PUF



Devices supporting PUF generated keys.

Flash with PUF Secured TEE



(green)	**PUF encrypted TEE**
(red)	**1st Stage Bootloader**
(purple)	**2nd Stage Bootloader**
(dark grey)	**Kernel/REE**

PUF Encrypted TEE Header		
OFFSET	**SIZE**	**DESCRIPTION**
0x0000	64 bytes	SHA-512 Hash of TEE
0x0040	4 bytes	TEE Magic Number
0x0044	4 bytes	TEE Manufacturer
0x0048	4 bytes	TEE Version Number
0x004C	4 bytes	Encryption Routine
0x0050	8 bytes	Offset to Bootloader
0x0054	424 bytes	Padding

Benefits of PUF Secured TEE

- Each device is encrypted with a different key, making information about one device useless for another.
- No one, even the manufacturer, knows what key each device is using.
- TEE measurement information is protected and rewriteable, providing a mechanism for expansion and customization of the TEE.
- Platform will support any TEE implementation available.

Applications of PUF Secured TEE

- This technology has several possible applications, including:
  - Handheld Cellular Devices
  - Femtocell Devices
  - Healthcare systems
  - Automotive systems
  - Handheld communications
  - Wireless Technologies
- Together with TEEs, this can provide protection for software applications, such as:
  - Banking Transactions
  - Mobile payment systems
  - Corporate VPN
  - NFC
  - Sensitive information transactions
  - Authentication
  - Remote attestation
  - Mobile Trusted Modules (mTPM)
  - OTA updates
  - Automotive firmware updates

Future Work

- Development of FPGA-based PUF.[1]
- Acquire Xilinx Zynq-7000 development kit with embedded dual-core ARM Cortex A9 processors.
- Import FPGA-based PUF onto platform and create cryptographic unit accessible from ARM CPUs.
- Utilizing ARM TrustZone technology, create TEE and store securely on Flash.
- Show ability to customize TEE to provide greater functionality.
- Port Android and incorporate into the system architecture.

[1] Already completed and published at HOST 2012.

Questions

?