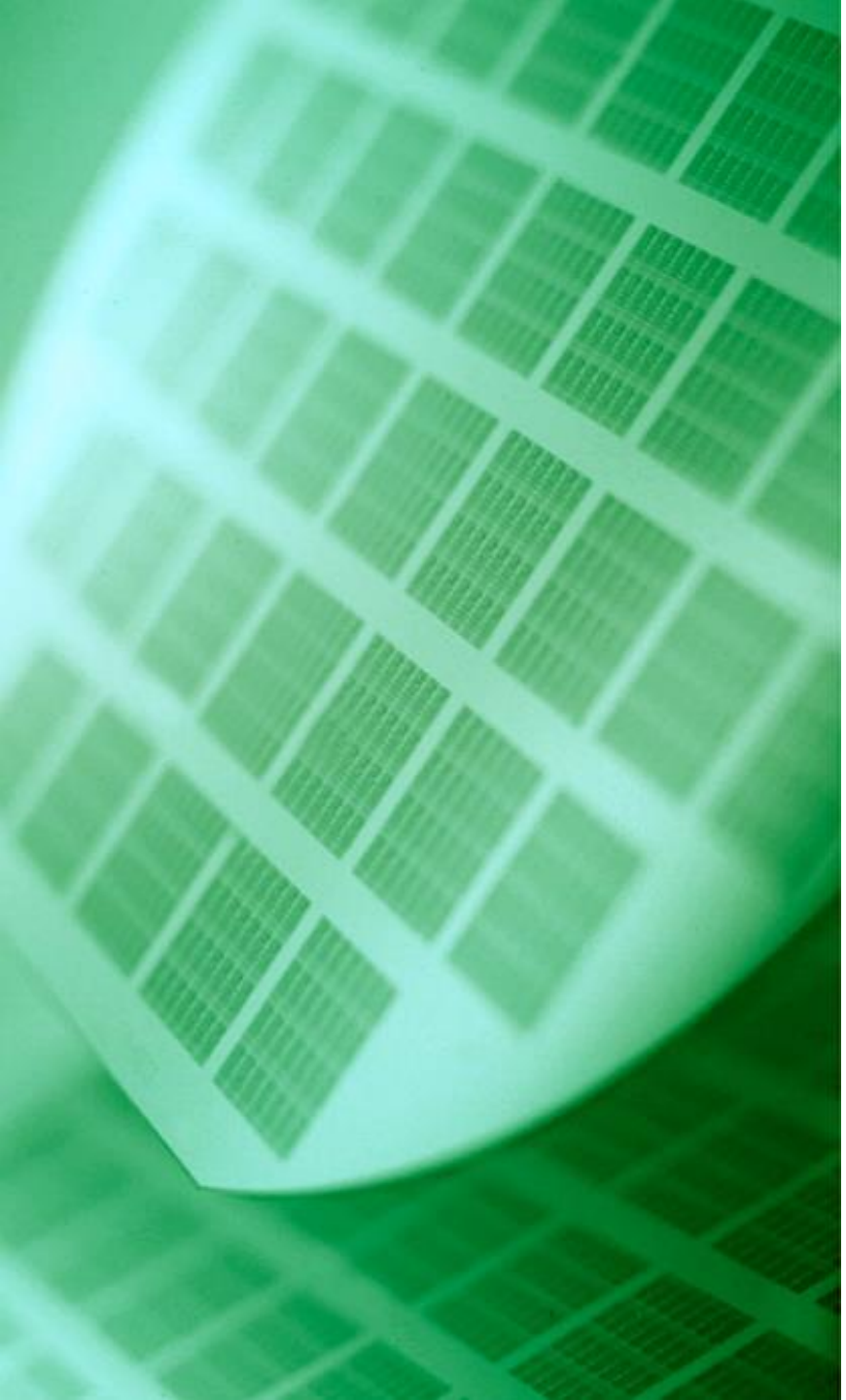


INTRINSIX

# Using RISC-V as a Security Processor for DARPA CHIPS and Commercial IoT

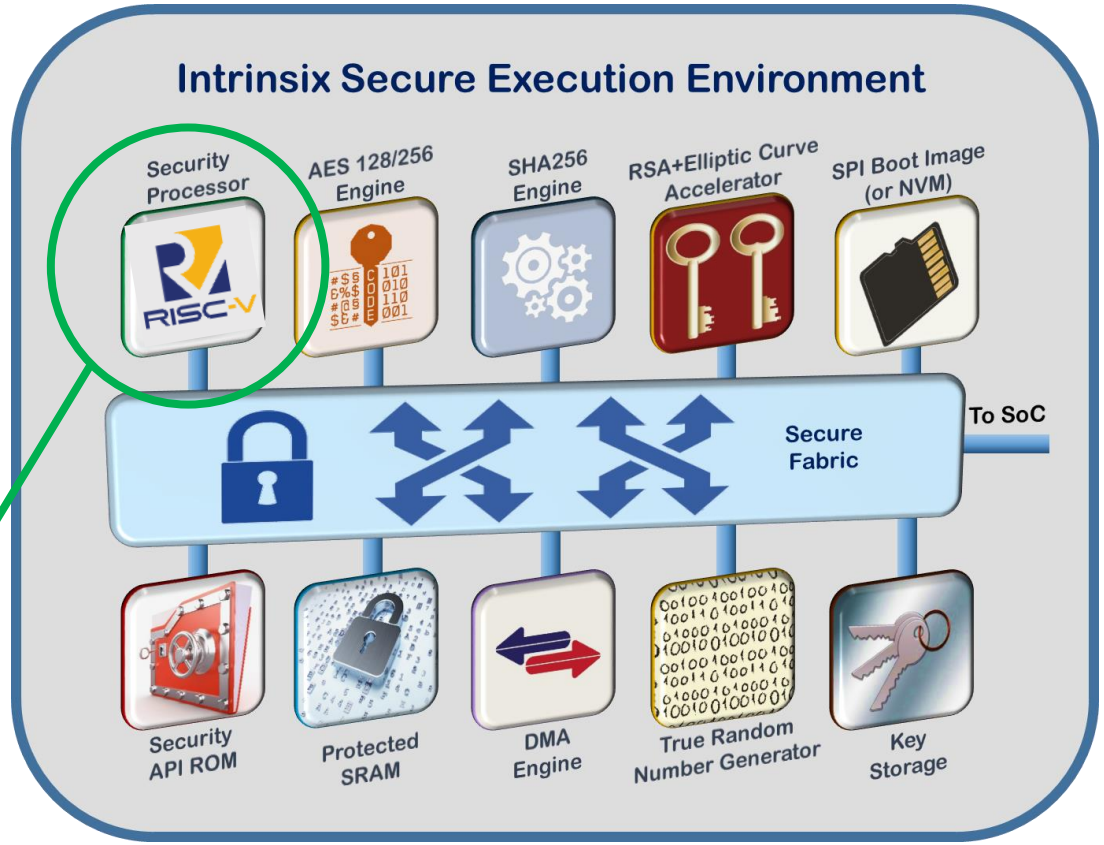
Mark Beal, CTO

November 29, 2017

# Enabling HW/SW Silicon IP Delivery

- Intrinsix Security & Crypto IP
  - Secure Execution Environment, Crypto Accelerators and Root-of-Trust
  - Secure Boot, Software Update, NSA Suite B Crypto, Key Management, etc
- This talk is not about:
  - RISC-V Security Extensions
  - RISC-V Crypto Extensions
  - RISC-V RocketChip

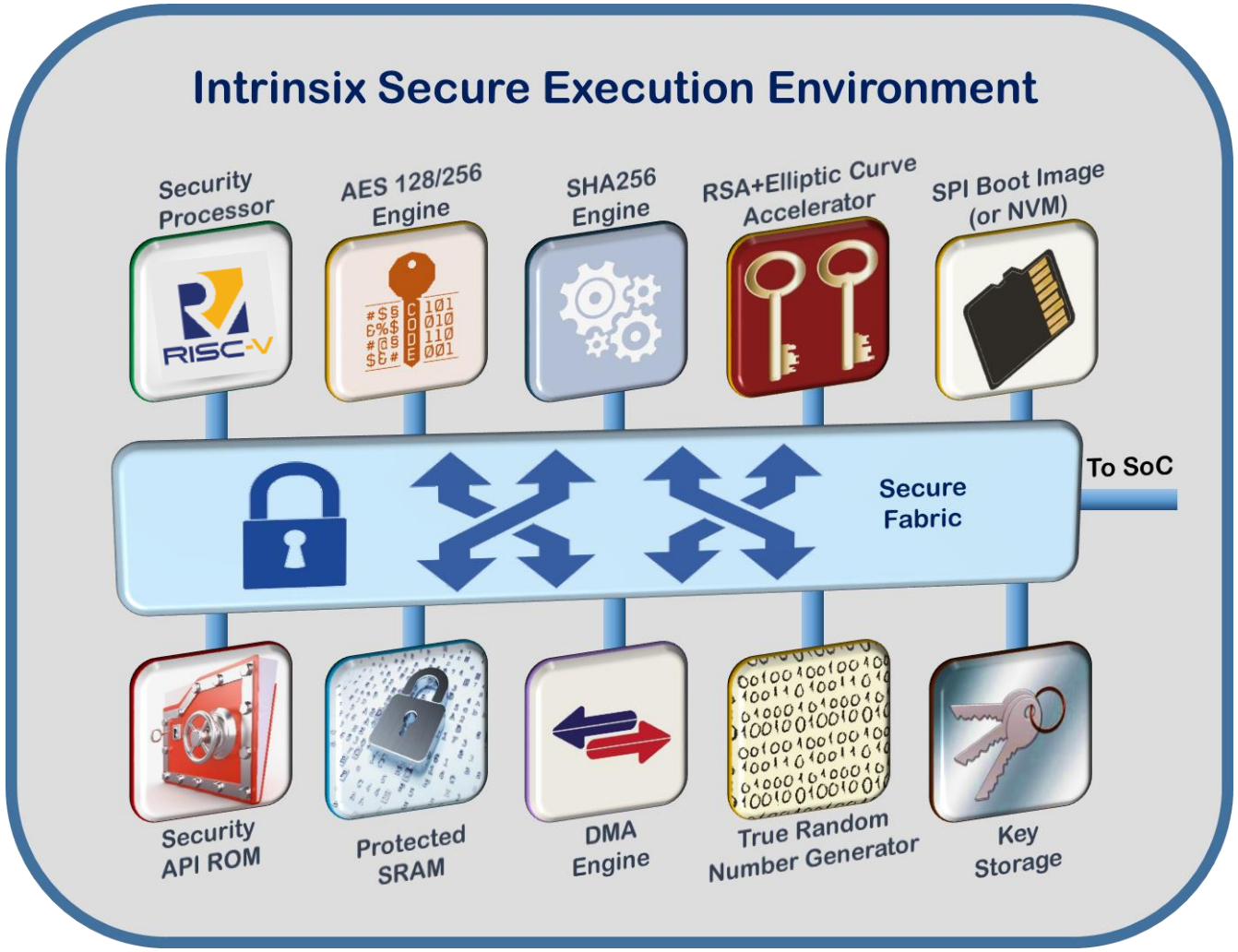
  (although these are great projects)
- This talk is about:
  - Leveraging RISC-V to deliver integrated HW/SW Silicon IP
  - RISC-V in a Security Block for DoD & IoT

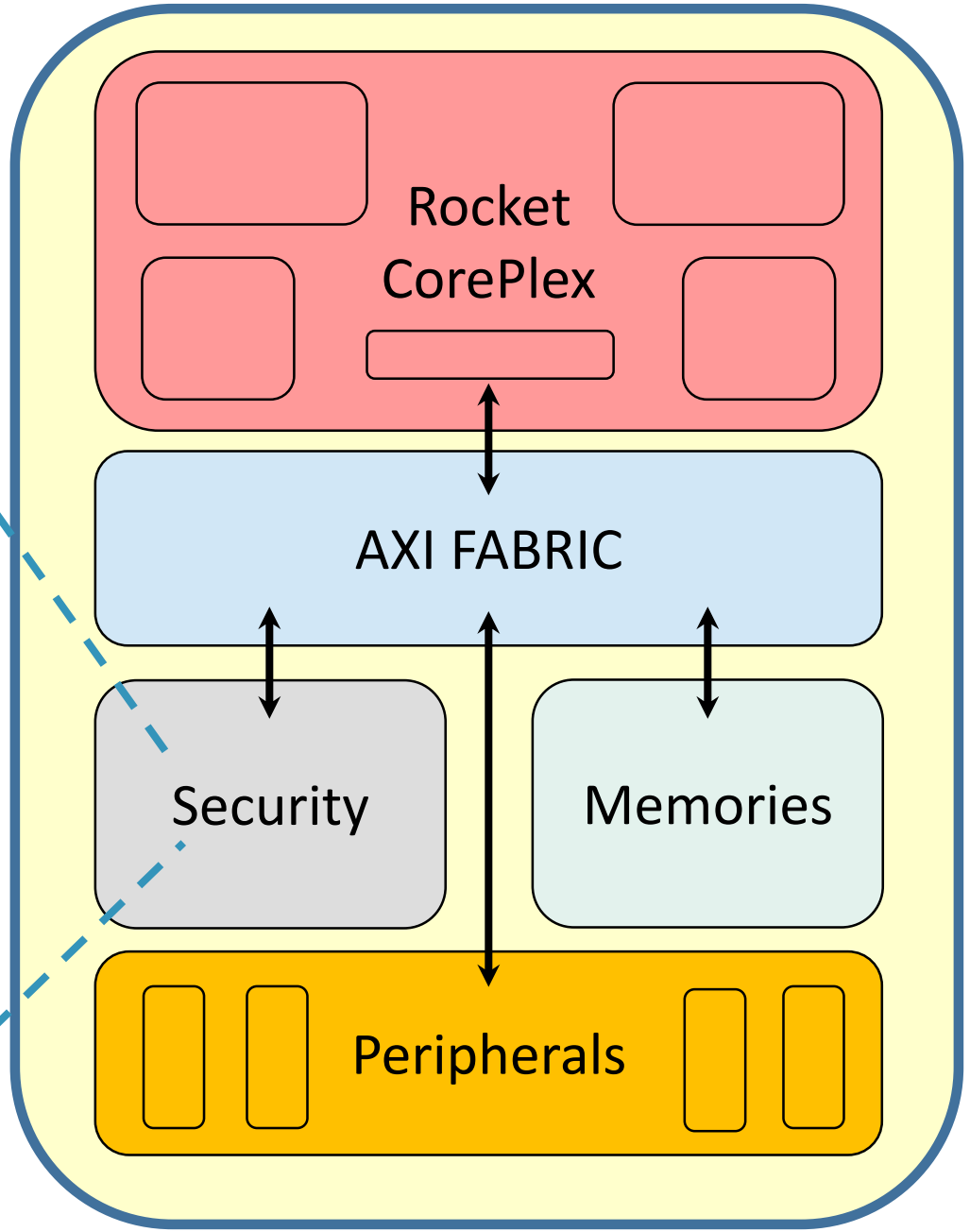


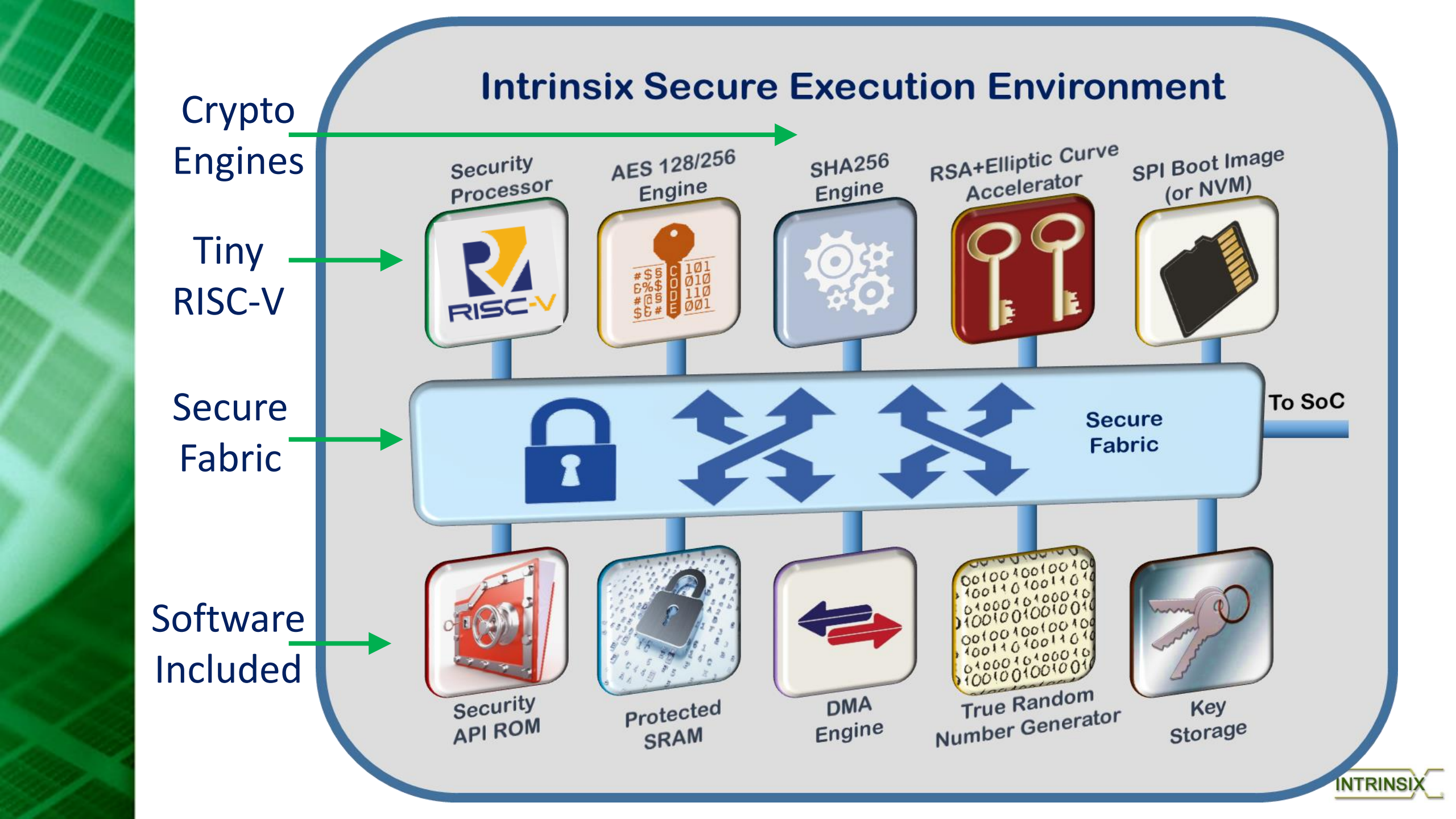

INTRINSIX

# IoT Edge Device SoC

## Security Subsystem

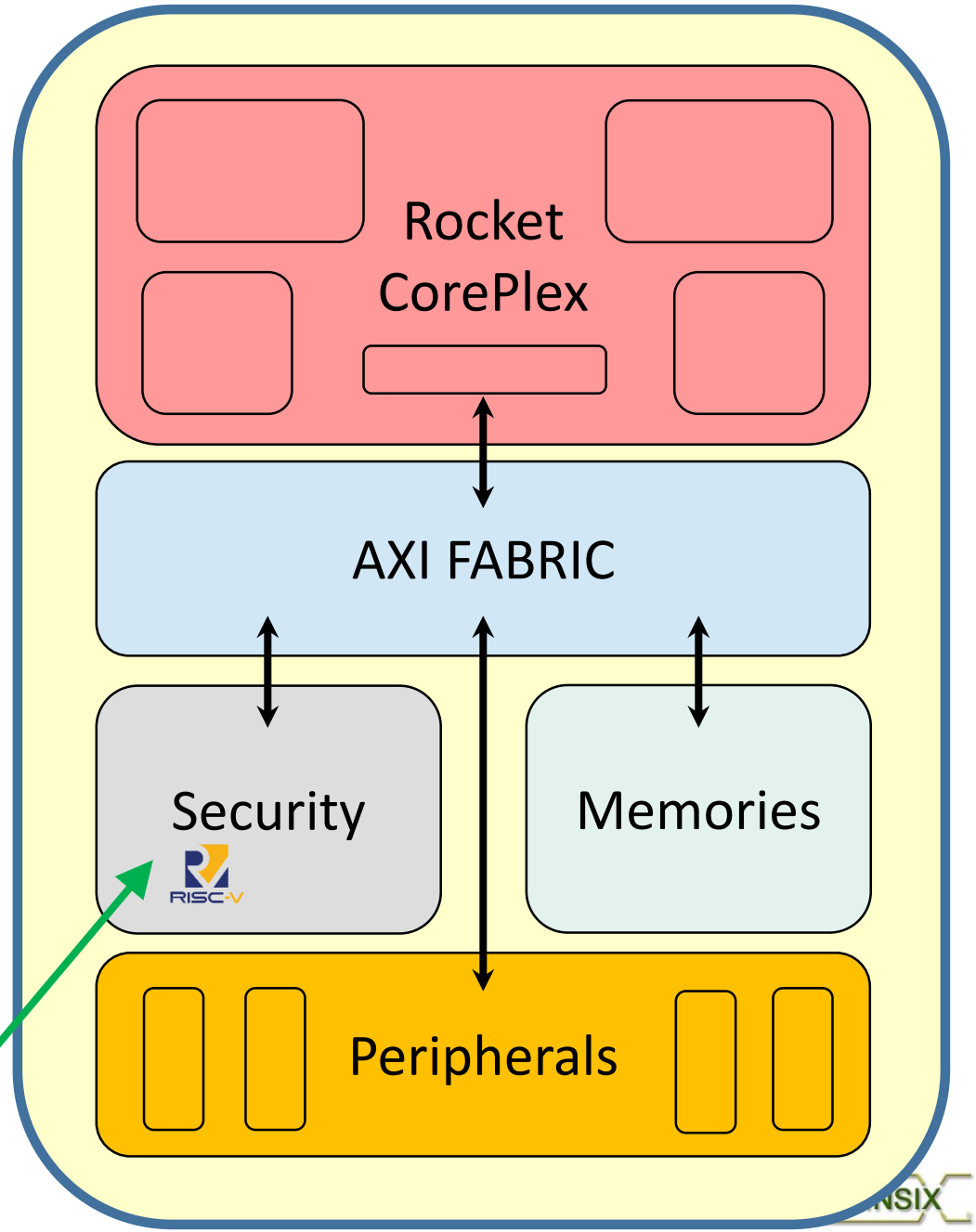


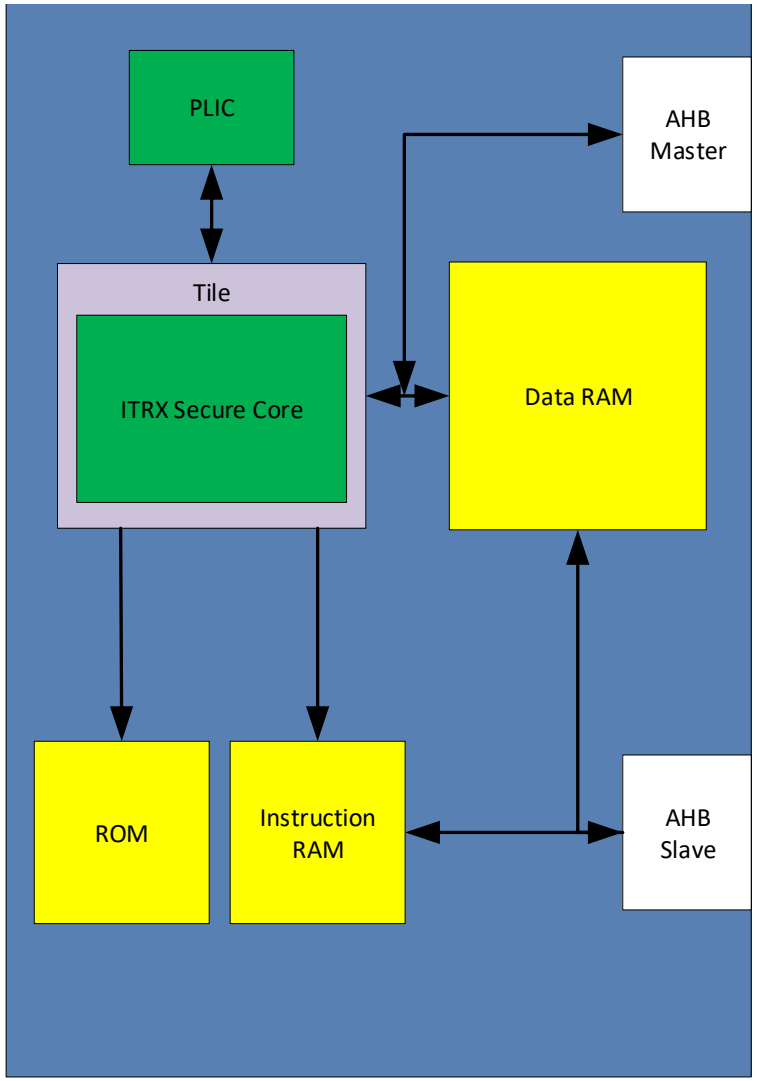

Crypto Engines
Tiny RISC-V
Secure Fabric
Software Included
Intrinsix Secure Execution Environment
Security Processor
AES 128/256 Engine
SHA256 Engine
RSA+Elliptic Curve Accelerator
SPI Boot Image (or NVM)
RISC-V
Secure Fabric
To SoC
Security API ROM
Protected SRAM
DMA Engine
True Random Number Generator
Key Storage
INTRINSIX

# Why add security CPU?

- Alternative is to have main CPU control the crypto engines, but security harder to assure
- Security processor benefits:
  - Isolated Execution Environment
  - Security software is locked down
  - Secrets are never held in main CPU
  - HW/SW interaction pre-verified for correctness, side channel resistance
  - Easier to verify separation between secure and non-secure actions
- Cost:  <1% silicon area (20K gates)



# Security Processor Implementation

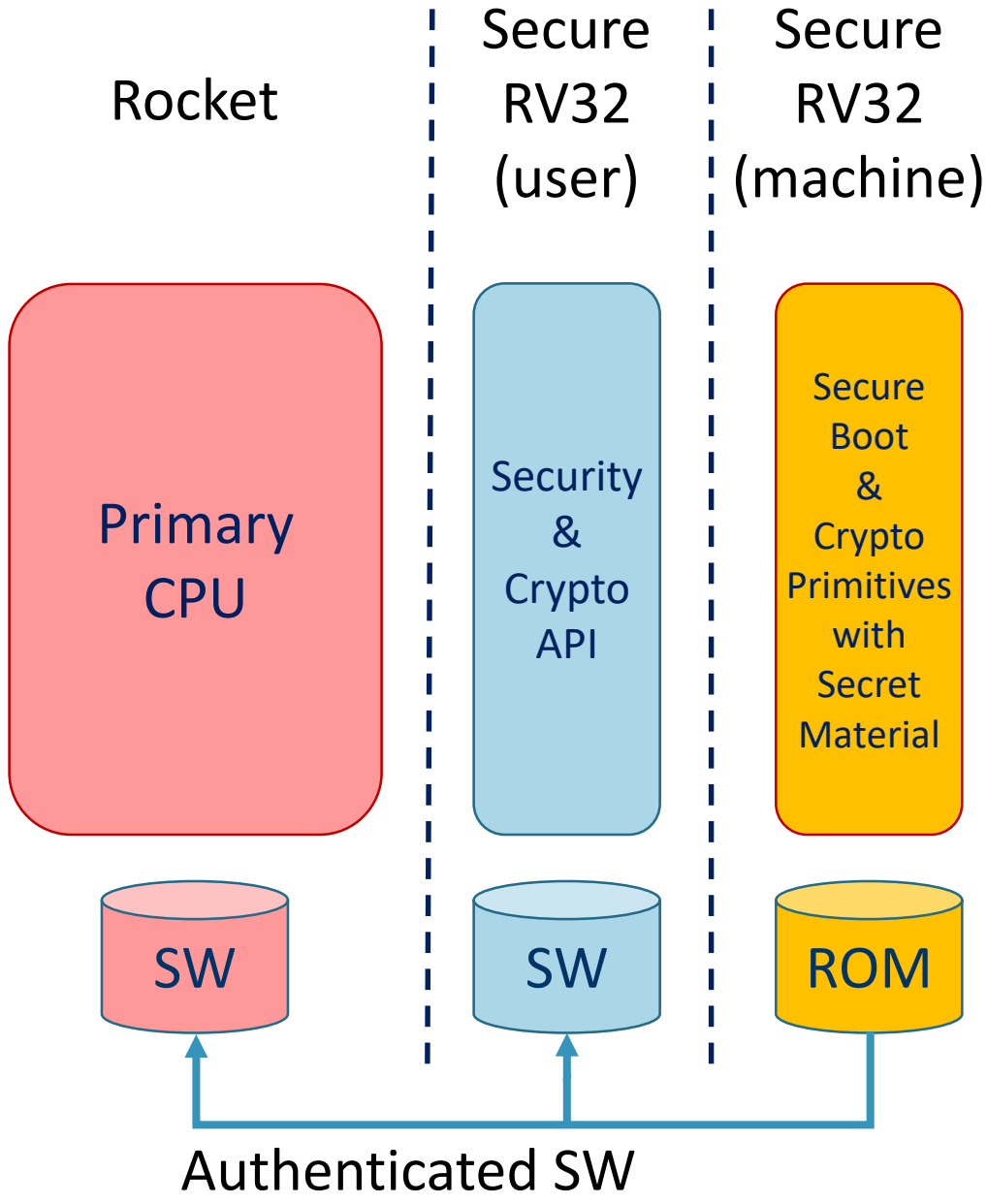



- RV32I + C ： Machine and User Modes
- 2-Stage Pipeline – Local ROM & RAM
- Local ROM contains initial boot code and sensitive security primitives
- Local ROM accessed in Machine Mode
- ROM code assures clean exit from security primitives, overwrites regs, etc
- Local IRAM can only hold signed code, fetch-only, locked after authentication
- Local IRAM access in User Mode, contains high level security protocols & functions

INTRINSIX

# Security Partitioning

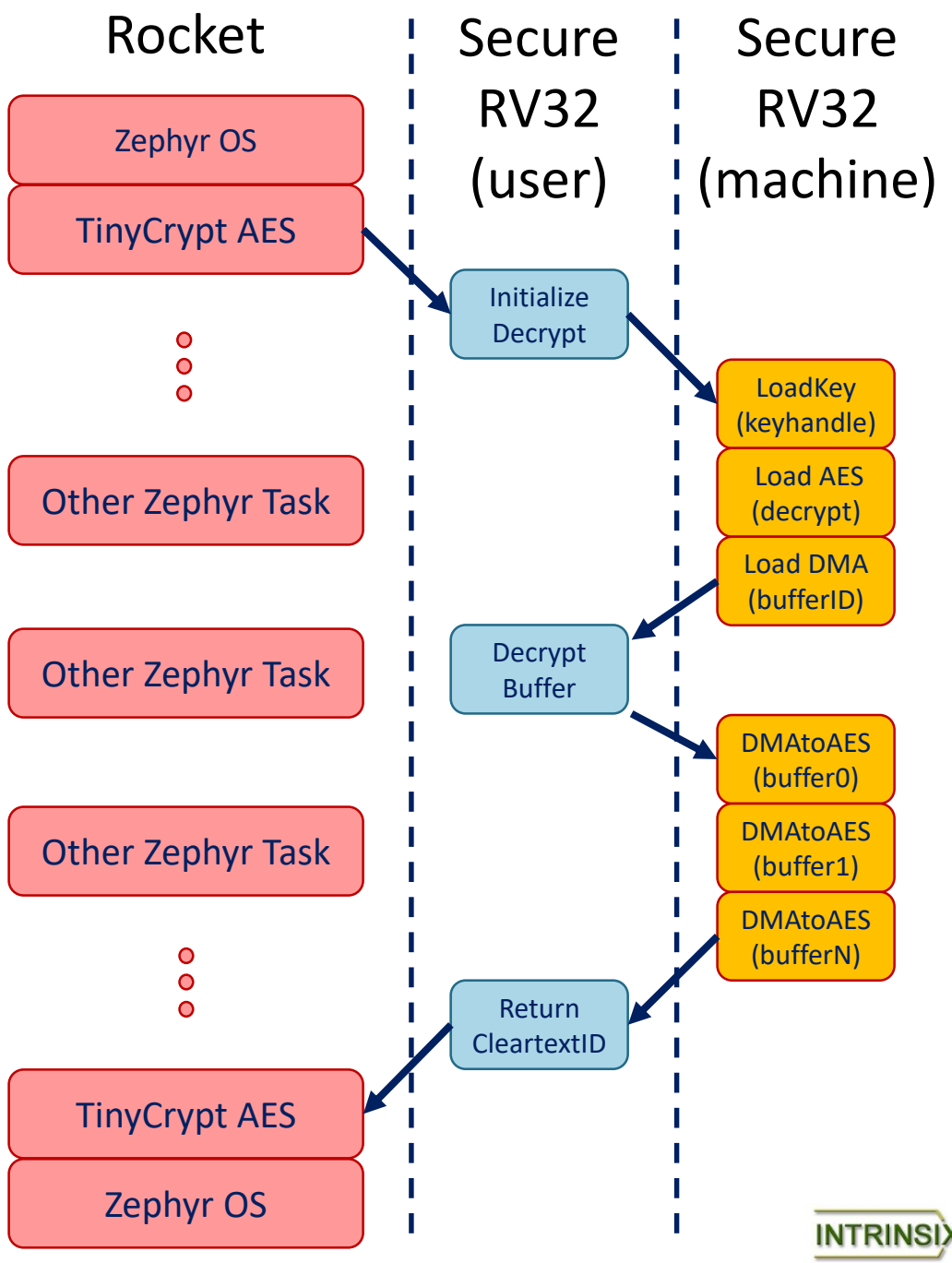
- Rocket is primary CPU
  - Runs OS, Runs Application
  - Linux (MMU) or IoT OS (MPU only)
  - OS & applications are authenticated
- Security RV32 in **User Mode**
  - Security algorithms encapsulated
  - Tested for correctness and side-channel
  - Signed firmware more flexible than ROM
  - Secret material not available to User Mode
- Security RV32 in **Machine Mode**
  - Only executes from hardwired ROM
  - Authenticates higher level firmware
  - Secret material handled by Machine Mode



# Example HW/SW API

- Zephyr OS running on Rocket
- Uses TinyCrypt as the Crypto API

- Replace TinyCrypt on Rocket with call to API running on Secure RV32
- Secure RV32 runs overall protocol in User Mode, making Machine Mode calls to security primitives
- Provides packaged “drop-in” hardware/software solution
- Can serve as Always-on processor




INTRINSIX

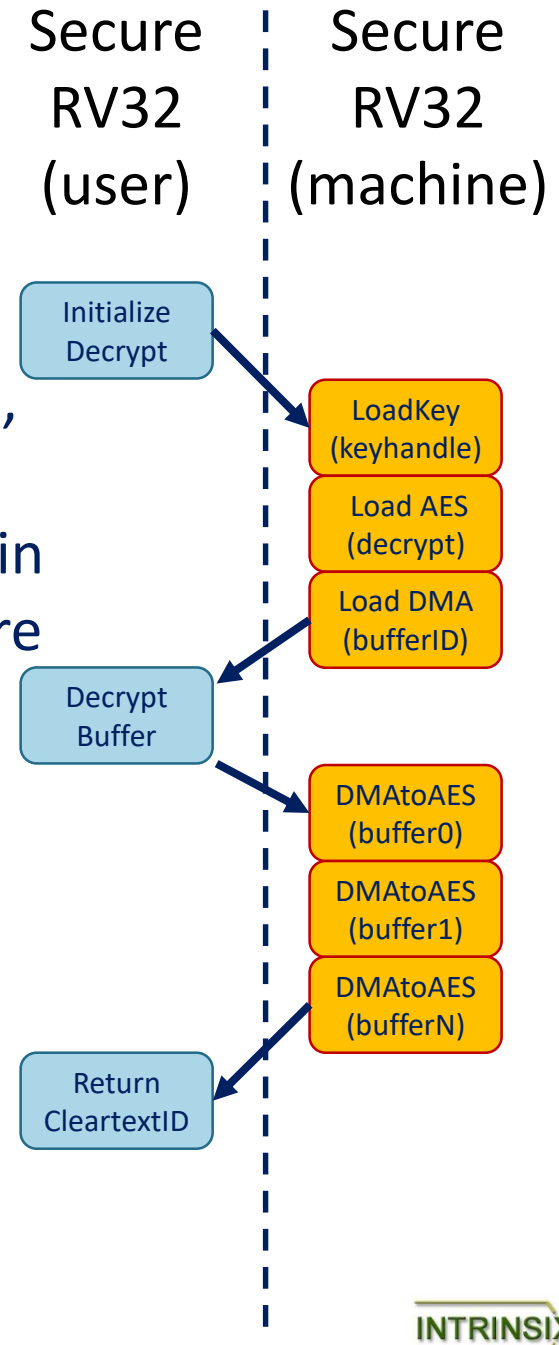
# Example HW/SW API

- Zephyr OS running on Rocket
- Uses TinyCrypt as the Crypto API

- Replace TinyCrypt on Rocket with call to API running on Secure RV32
- Secure RV32 runs overall protocol in User Mode, making Machine Mode calls to security primitives
- Provides packaged “drop-in” hardware/software solution
- Can also serve as always-on cpu

For tiny IoT devices, application code is authenticated within User Mode firmware

Eliminates big CPU, while maintaining IoT device security




INTRINSIX