Enclaves in Real-Time Operating Systems

Alex Thomas, Stephan Kaminsky, Dayeol Lee, Dawn Song, Krste Asanovic

Motivation

- Influx of real-time devices
  - Processing sensitive information
- Edge Computing
- Increased third-party applications on real-time devices
  - Crypto libraries, IDS, etc.
- Attacks targeting these devices
  - Tesla Attack through Connman [1]

Existing Solutions

- RTOS Kernel Security
  - Not reliable [2]
- Memory Protection Units (MPUs)
  - Doesn’t protect adversary RTOS
- Software-Fault Isolation
  - Performance Overhead

Goals

1. Strong isolation
2. Negligible performance overhead
3. Protection against adversary RTOS

Solution: Trusted Execution Environments

Trusted Execution Environments (TEE)

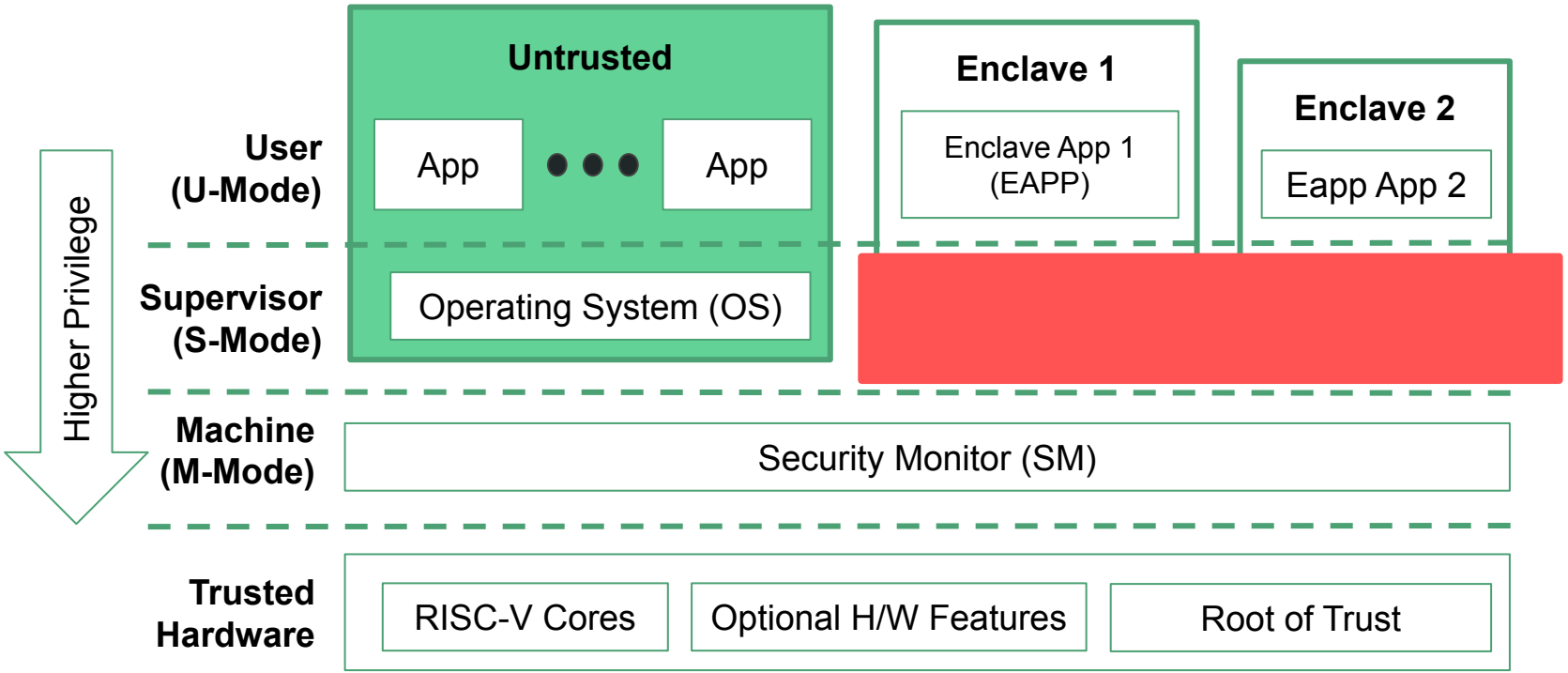
- Enables secure computation and isolation
  - Even from privileged OS!
- Must be suitable in an embedded system context..
  - No expensive hardware
- Dynamic TEE creation
  - Dynamic installation of 3rd party apps
- Multi-isolation
  - Isolate between tasks

TEE/Enclave Backends

- Intel Software Guard Extensions (SGX)
  - Expensive hardware (i.e. Memory Encryption Engine)
  - Required VM Support
- ARM TrustZone
  - Single zone architecture
- MultiZone
  - No dynamic TEE creation

Keystone

- Open-source framework to create customizable TEEs
- Based on RISC-V architecture
  - Isolation via PMP registers
- No reliance on VM
  - Easy to remove S-mode component
- Dynamic multi-TEE creation
- Software Encryption/Integrity


Untrusted
Enclave 1
Enclave 2
User
(U-Mode)
App
App
Enclave App 1
(EAPP)
Eapp App 2
Supervisor
(S-Mode)
Operating System (OS)
Higher Privilege
Machine
(M-Mode)
Security Monitor (SM)
Trusted
Hardware
RISC-V Cores
Optional H/W Features
Root of Trust

FreeRTOS

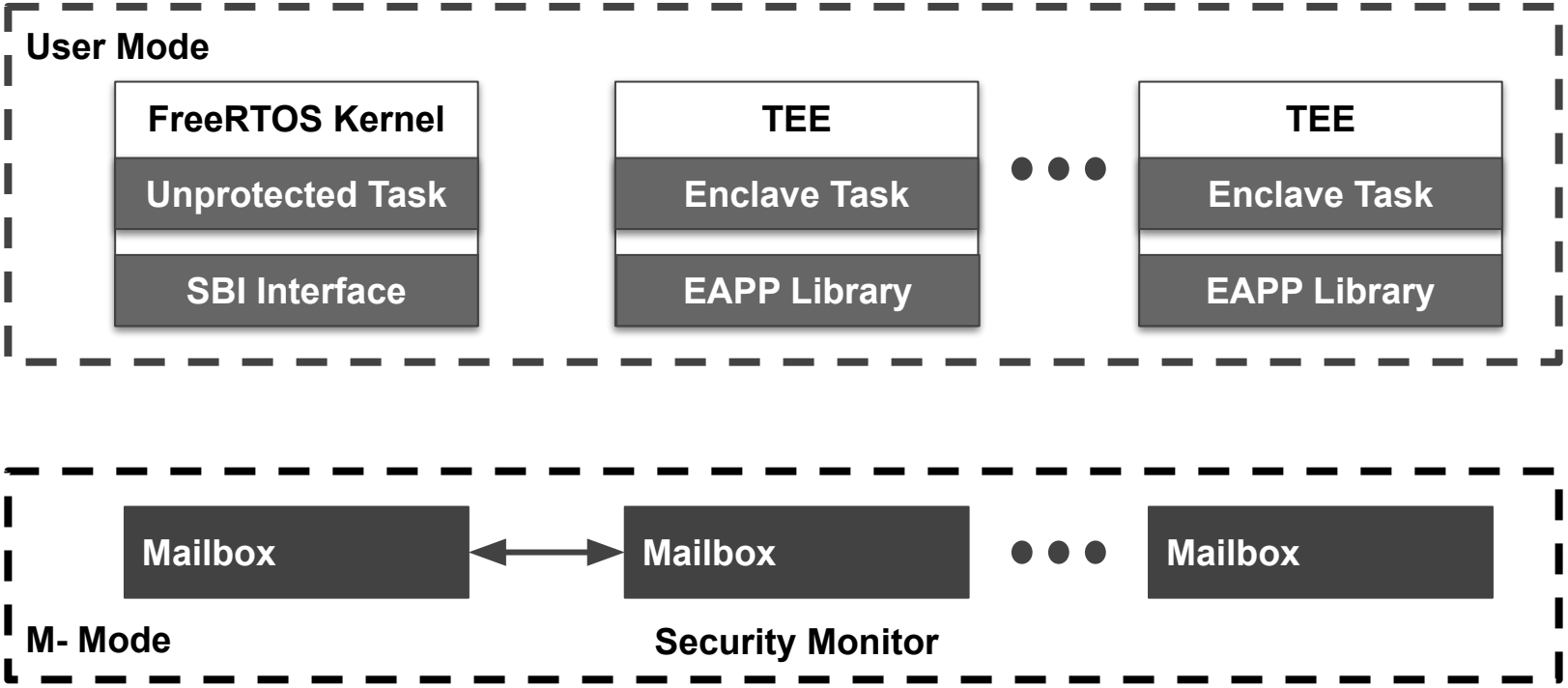
- Open-source!
- Popular RTOS owned by Amazon
  - Libraries to interface with AWS
- Small Footprint
  - Kernel is only 3 files
- Add-on libraries
  - TCP/IP
  - I/O

Keystone + FreeRTOS

- **Keystone isn’t a scheduler**
- We still need an RTOS
  - Take away privileges
- Solution
  - Combine FreeRTOS + Keystone

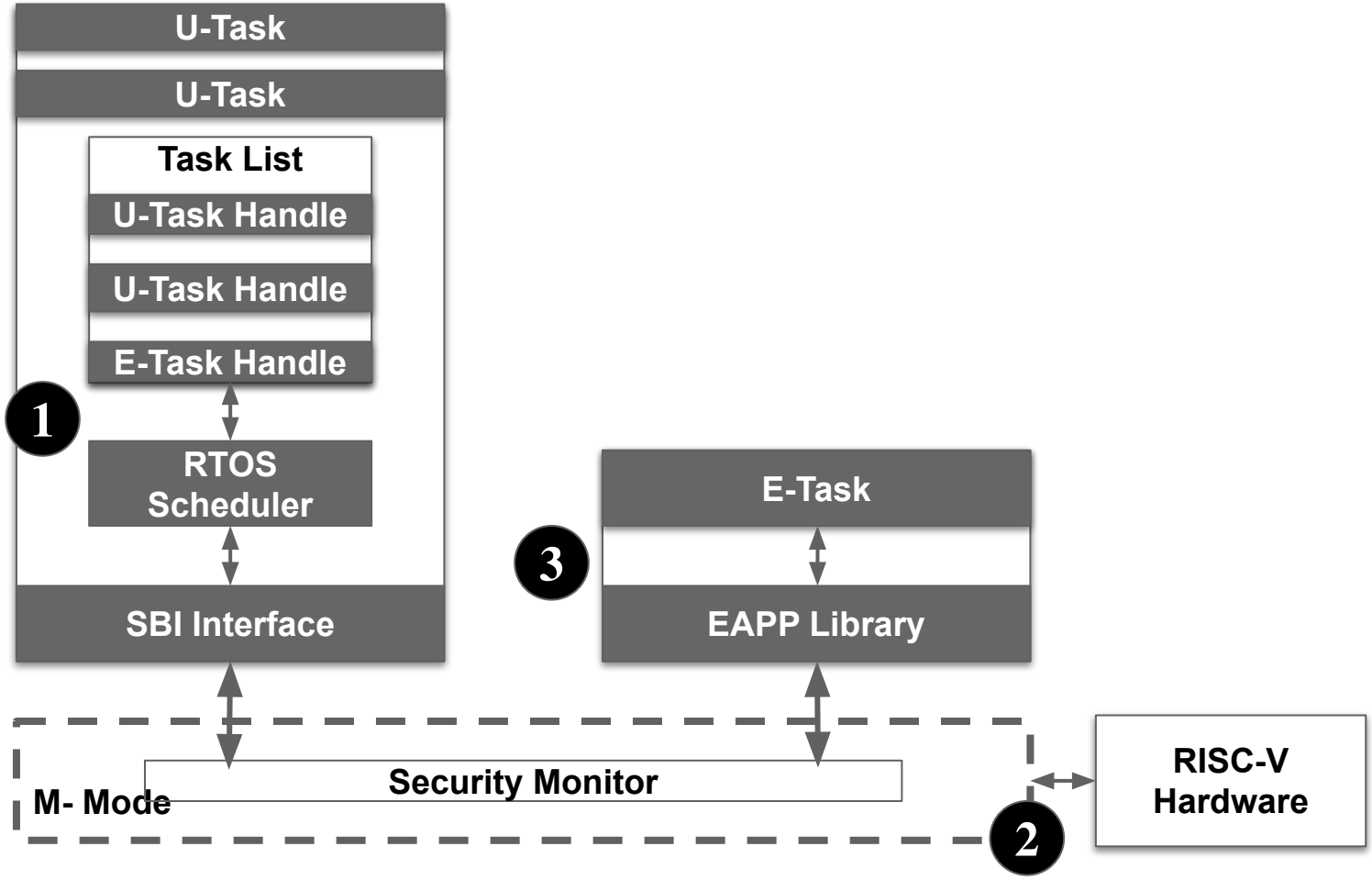
FreeRTOS Module -- ERTOS

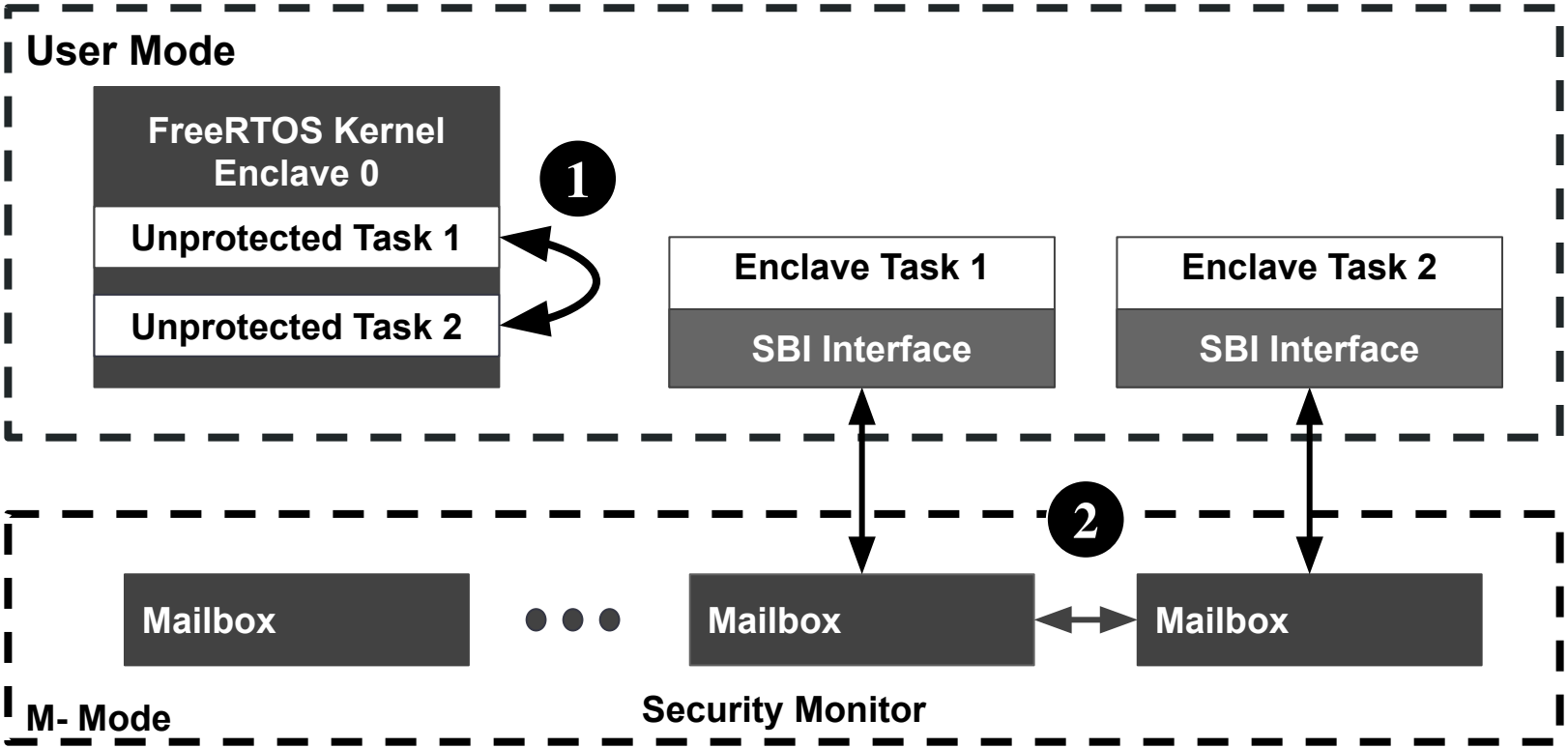
- Created a module in FreeRTOS
  - APIs to allow enclave creation, execution, etc.
- Used Keystone as a TEE backend
  - Security Monitor manages enclaves
- FreeRTOS protected by an enclave
  - User-mode RTOS
  - Schedules tasks
- Tasks can be..
  - Unprotected
  - Secure (TEE)


User Mode
FreeRTOS Kernel
Unprotected Task
SBI Interface
TEE
Enclave Task
EAPP Library
TEE
Enclave Task
EAPP Library
Mailbox
Mailbox
Mailbox
M- Mode
Security Monitor

FreeRTOS Enclave

- Allowed to signal to SM to create, execute, or delete enclaves
  - Restricted for enclave tasks
- All interrupts to an enclave task switch to RTOS enclave
  - Mitigate DoS

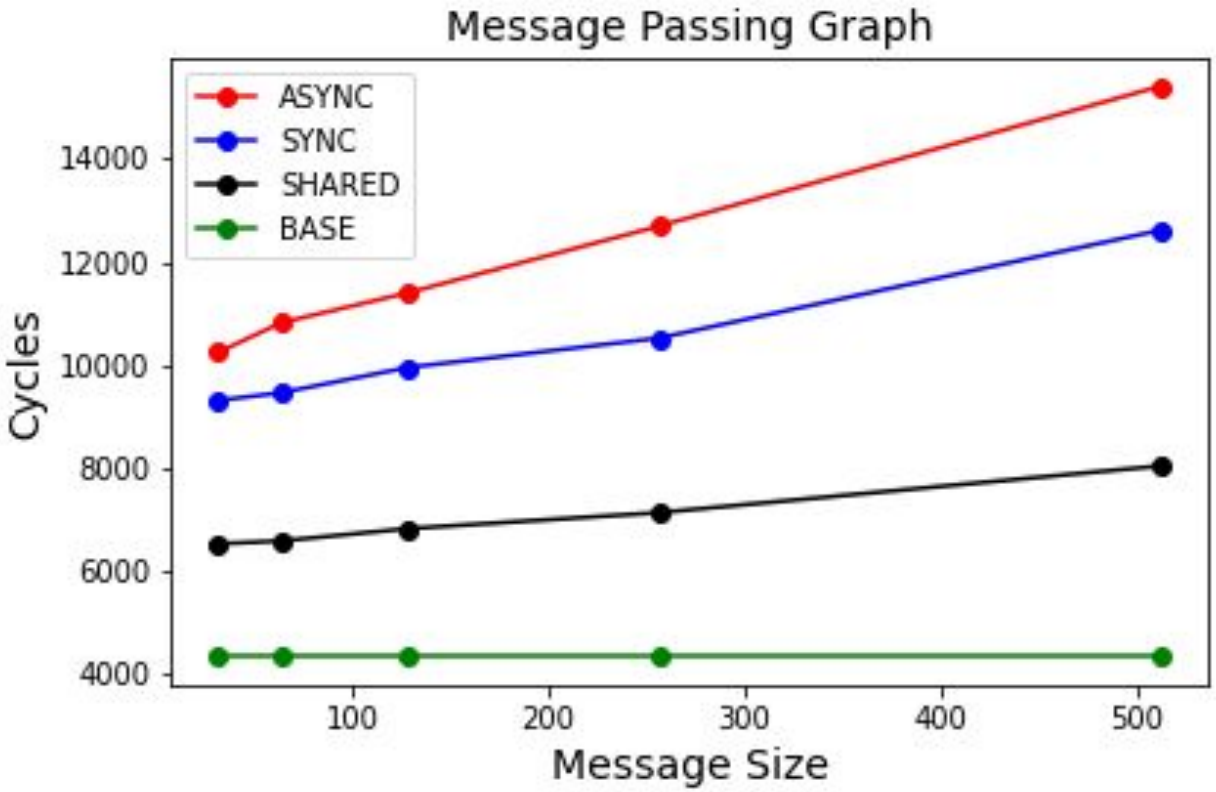

U-Task
U-Task
Task List
U-Task Handle
U-Task Handle
E-Task Handle
1
RTOS Scheduler
SBI Interface
E-Task
3
EAPP Library
Security Monitor
M- Mode
2
RISC-V Hardware

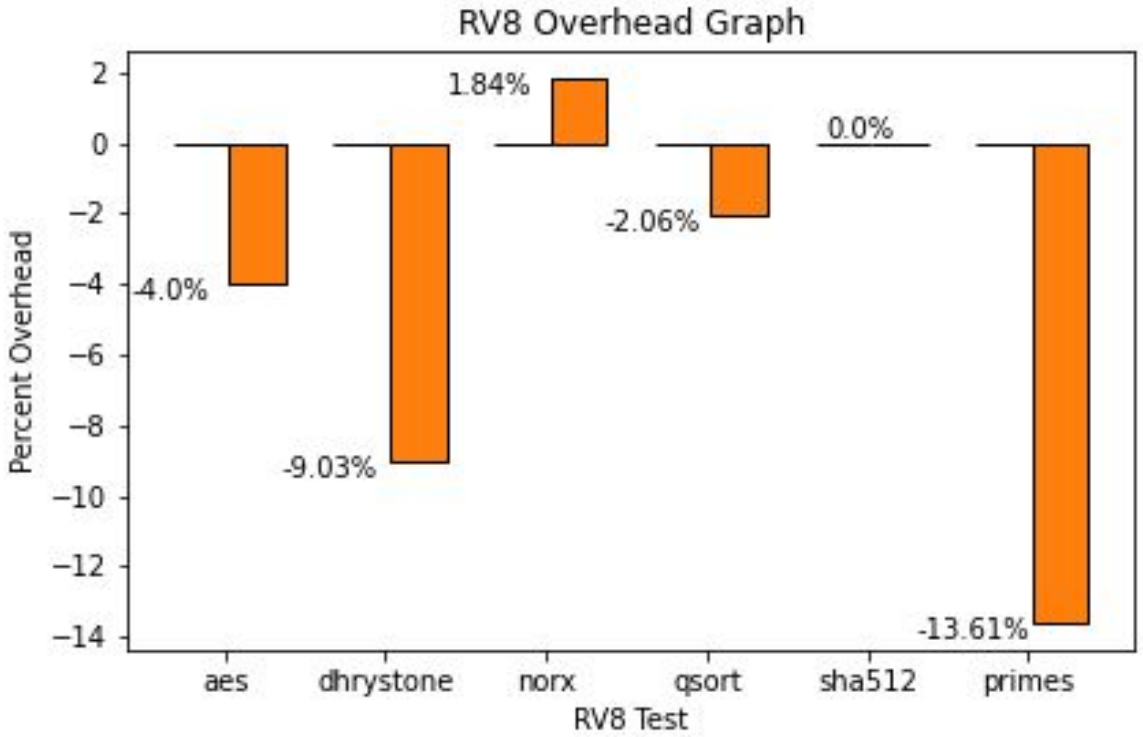

User Mode
FreeRTOS Kernel
Enclave 0
Unprotected Task 1
Unprotected Task 2
1
Enclave Task 1
SBI Interface
Enclave Task 2
SBI Interface
2
Mailbox
Mailbox
Mailbox
M- Mode
Security Monitor

Results

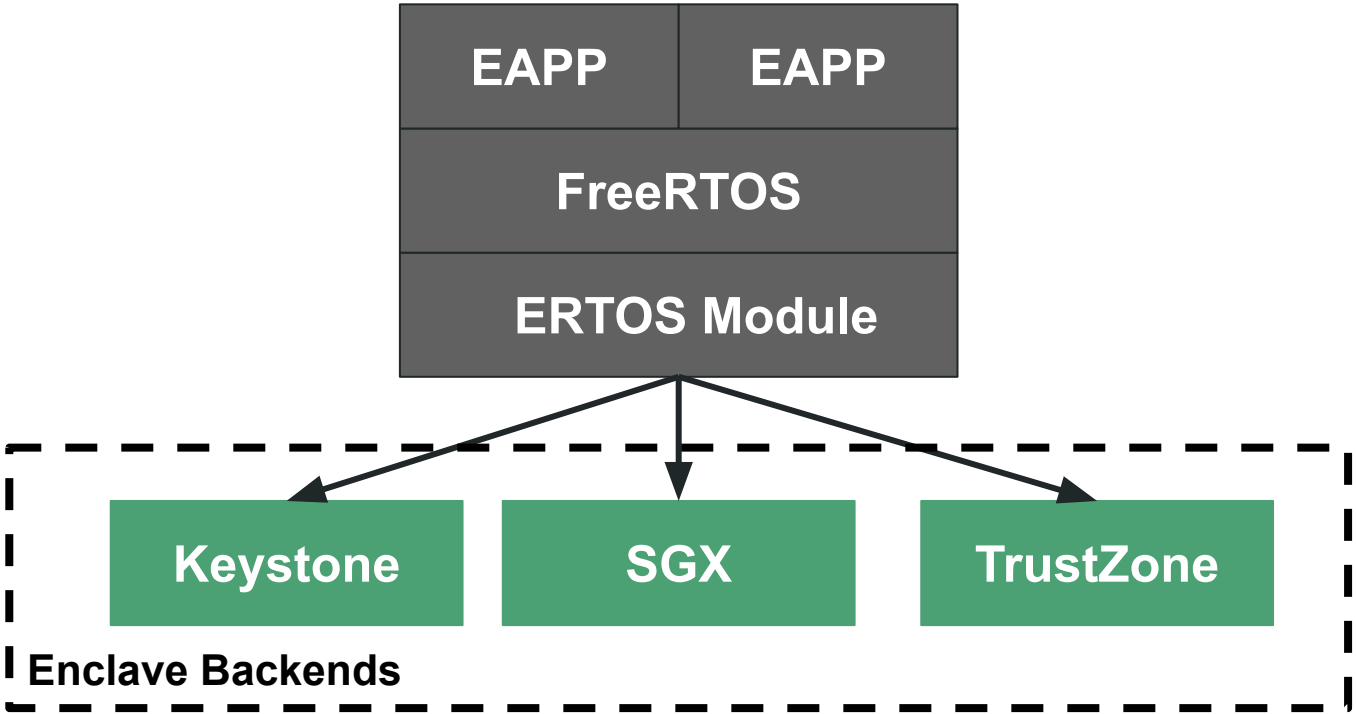
Message Passing Modes

- Enclave Tasks
  - Asynchronous Messages via Mailbox
  - Synchronous Message Passing
    - Single Copy
  - Shared buffer
    - Consumes PMP
- Normal Tasks
  - Zero-copy Queue
    - Between non-secure tasks


Message Passing Graph
ASYNC
SYNC
SHARED
BASE
Cycles
14000
12000
10000
8000
6000
4000
100
200
300
400
500
Message Size


RV8 Overhead Graph
Percent Overhead
2
0
−2
−4
−6
−8
−10
−12
−14
-4.0%
-9.03%
1.84%
-2.06%
0.0%
-13.61%
aes
dhrystone
norx
qsort
sha512
primes
RV8 Test

Future Work


EAPP
EAPP
FreeRTOS
ERTOS Module
Keystone
SGX
TrustZone
Enclave Backends

Thank you!

Contact:

alexthomas@berkeley.edu

Bibliography

[1] R. Weinmann & B. Schmotzle “TBONE - A zero-click exploit for Tesla MCUs” https://kunnamon.io/tbone/tbone-v1.0-redacted.pdf

[2] G. Mullen and L. Meany, "Assessment of Buffer Overflow Based Attacks On an IoT Operating System," 2019 Global IoT Summit (GIoTS), 2019, pp. 1-6, doi: 10.1109/GIOTS.2019.8766434.