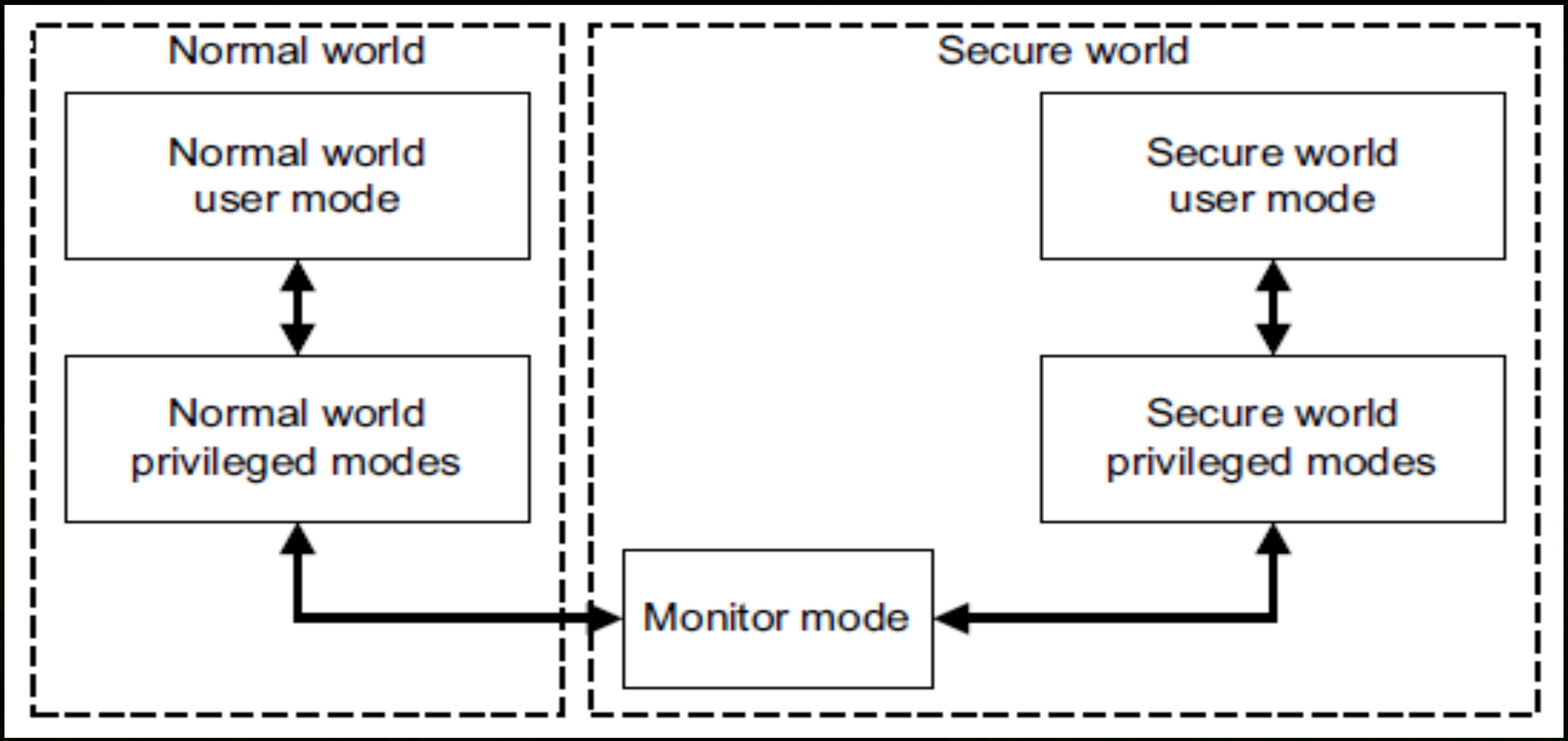
TLK:
A FOSS Stack for Secure Hardware Tokens

Hadi Nahari
Chief Security Architect
NVIDIA

TrustZone®


Normal world
Normal world user mode
Normal world privileged modes
Secure world
Secure world user mode
Secure world privileged modes
Monitor mode

TrustZone® and Hardware-based TCB

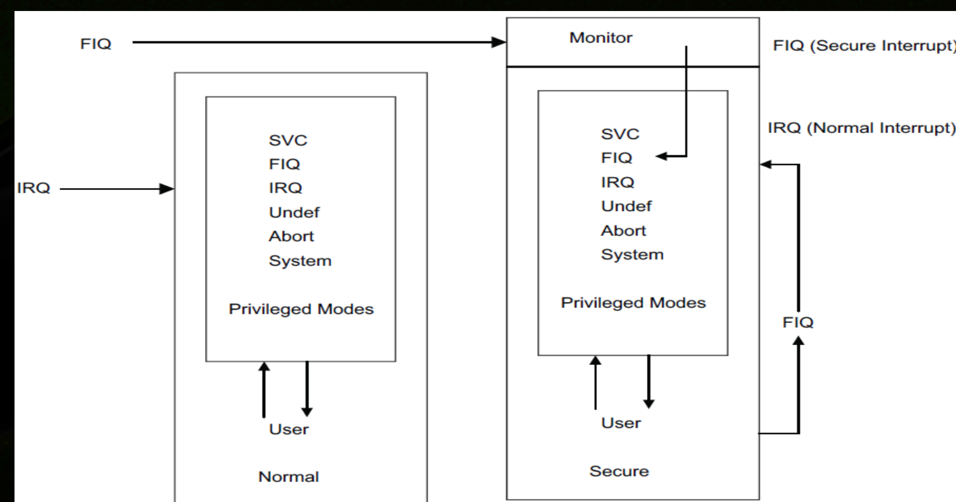
- Each of the physical processor cores provide two virtual cores
  - Secure (Secure World for the security subsystem)
  - Non-secure (Normal World for everything else)
- New core mode: Monitor Mode
  - A mechanism to context-switch between two states (secure ⇔ non-secure)
- A limited set of mechanisms to enter the Monitor Mode
  - S/W: SMC instruction from software
  - H/W: IRQ, FIQ, external (prefetch, Data) aborts

TrustZone® Processor Architecture

- The NS bit in the SCR in CP15 indicates which state (aka “world”) the processor is currently in
  - NS = 1 ➪ processor is in non-secure state
  - NS = 0 ➪ processor is in secure state
  - SCR can only be accessed in secure state
- Monitor Mode is always running in secure state
  - regardless of the value of NS bit

TrustZone® Secure Interrupts

- ARM recommendation:
  - IRQ for normal world
  - FIQ for secure world
- IRQ and FIQ can be directly trapped to Monitor Mode
- Vector Based Address Register
  - For non-secure, secure, and monitor



TEE: Trusted Execution Environment

- A carve-out within Application Processor (AP)
  - Allows for running a trusted piece of code
  - Provides hardware-based isolation
  - Enables privileged access to device resources (e.g. memory, hardware crypto accelerator(s), etc.)
- ARM TrustZone® is *one* way to implement a TEE
  - Is not the only way

TEE Use Cases

- Secure hardware tokens
- Mobile payment
- BYOD
- Runtime integrity verification
- Trusted user interface
- Remote enablement/disablement
- Automotive (trust vs. safety)
- Secure isolation, Remote attestation
- DRM, HDCP, secure NFC in P2P mode
- Any other operation that requires verifiable trust

GlobalPlatform™ and TEE

- TEE WG of GlobalPlatform™ standardizes the TEE & its APIs
- GlobalPlatform™*
  - GlobalPlatform works across industries to identify, develop and publish specifications which facilitate the secure and interoperable deployment and management of multiple embedded applications on secure chip technology. GlobalPlatform Specifications enable trusted end-to-end solutions which serve multiple actors and support several business models.
  - (source: http://www.globalplatform.org/aboutusmission.asp)

GLOBALPLATFORM™

TEE Ecosystem

- Main TEE ecosystem roles/entities
  - Chip vendor
  - Device vendor (OEM/ODM)
  - TEE stack vendor
  - TSM (Trusted Service Manager)
  - SP (Service Provider)
  - TA (Trusted Application) provider
- Each entity has a specific role: defined by GlobalPlatform™
- TEE stack vendors usually play the TSM role as well

TLK: Trusted Little Kernel

- What
  - An open source and royalty free software (i.e. FOSS) stack for TrustZone® to accelerate the adoption of hardware-based security for SoC, device, system, and service providers
- Why
  - Kerckhoffs's Desiderata: enabling a more secure ecosystem
  - Allow unencumbered pre-silicon, partner development and verification efforts
  - Existing TrustZone® software stacks facing variety of challenges supporting all requirements of our partners, including Defense & Intelligence Communities

Foundation

- TLK is based on LK (Little Kernel)
- LK
  - ~63 KLOC in C, with ARM emulation .bin ~22KB
  - Small, pre-emptive kernel
  - Supports Cortex-M3, Cortex-A8, AVR32, x86 SoC families
  - Supports multi-threading, IPCs, and thread scheduling
  - No TrustZone® features present
  - MIT/FreeBSD license
  - Designed, implemented and maintained by Travis Geiselbrecht, Dima Zavin, et al

Overview

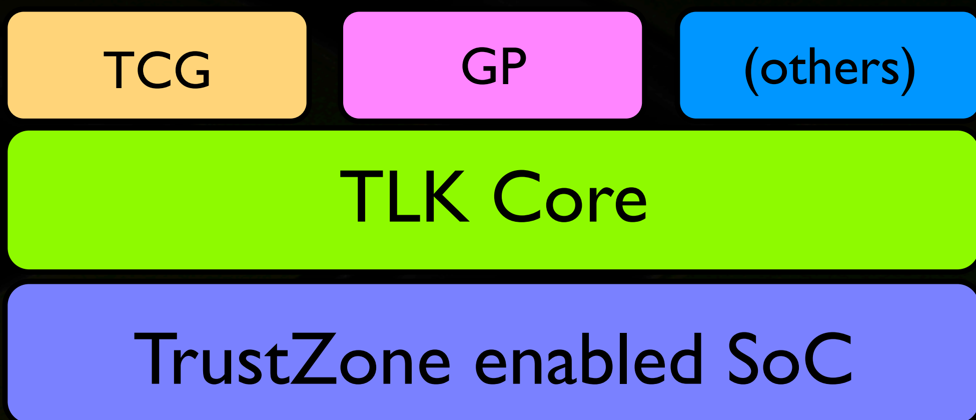
- TLK
  - ~23 KLOC in C
  - Supports multi-threading, IPC, thread scheduling
  - Implements TrustZone® features
  - Provides detailed documentation
  - Maintains MIT/FreeBSD license
- Not limited to Tegra SOCs

Design Criteria

- Open source
- Extensible
- Easy to learn
- Open tools (e.g. gcc)
- Interrupt, SMP, Secure timer
- Deferred startup of services
- Crypto ops
- Simulator (QEMU)

- Code
  - Clean
  - Small size
  - Well structured
- Existing security constructs
- Multiple security paradigms
  - GP
  - TCG
  - …

Componentized Architecture

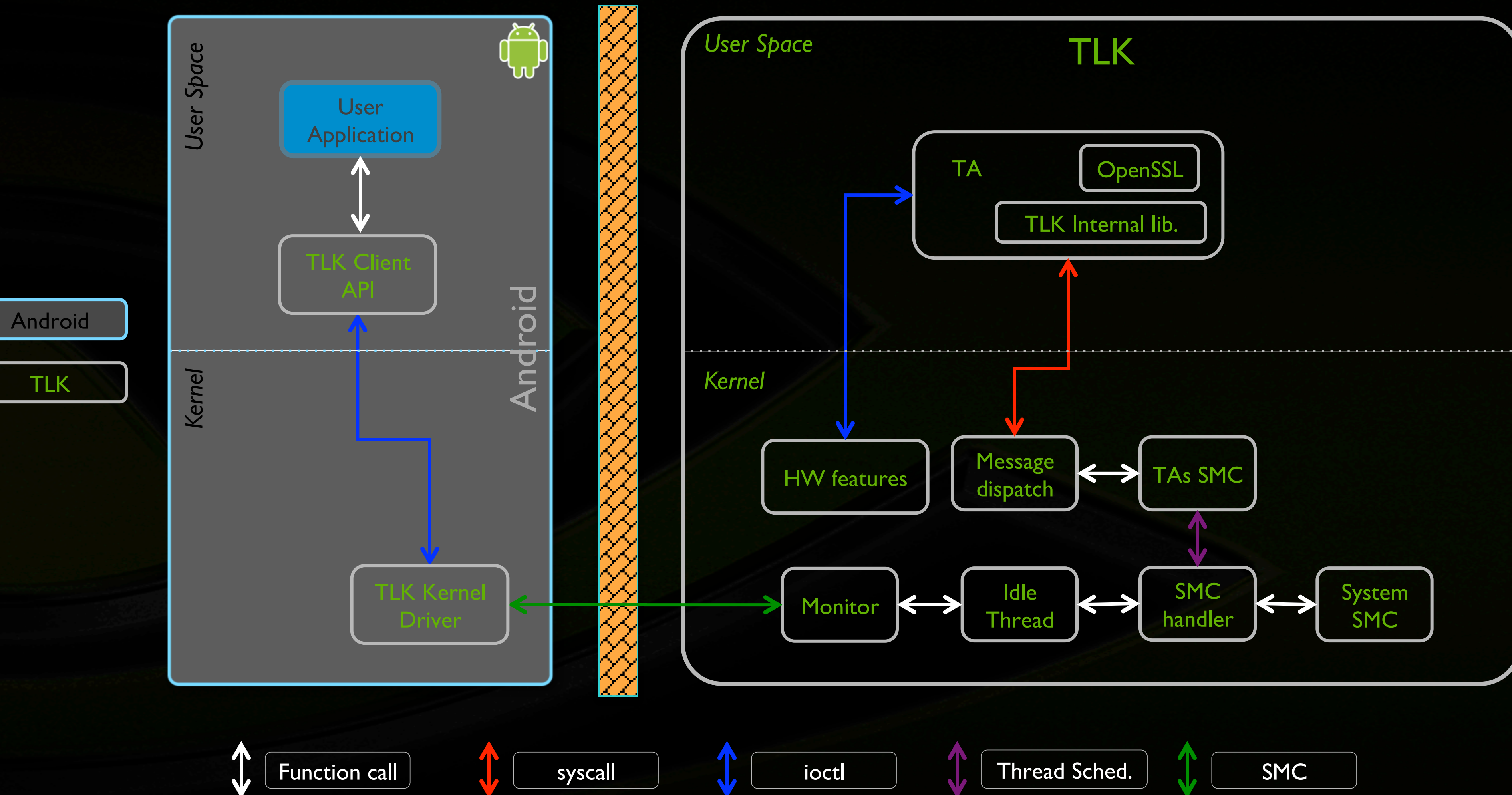


- Tegra
- Snapdragon
- OMAP
- Exynos
- NovaThor
- …

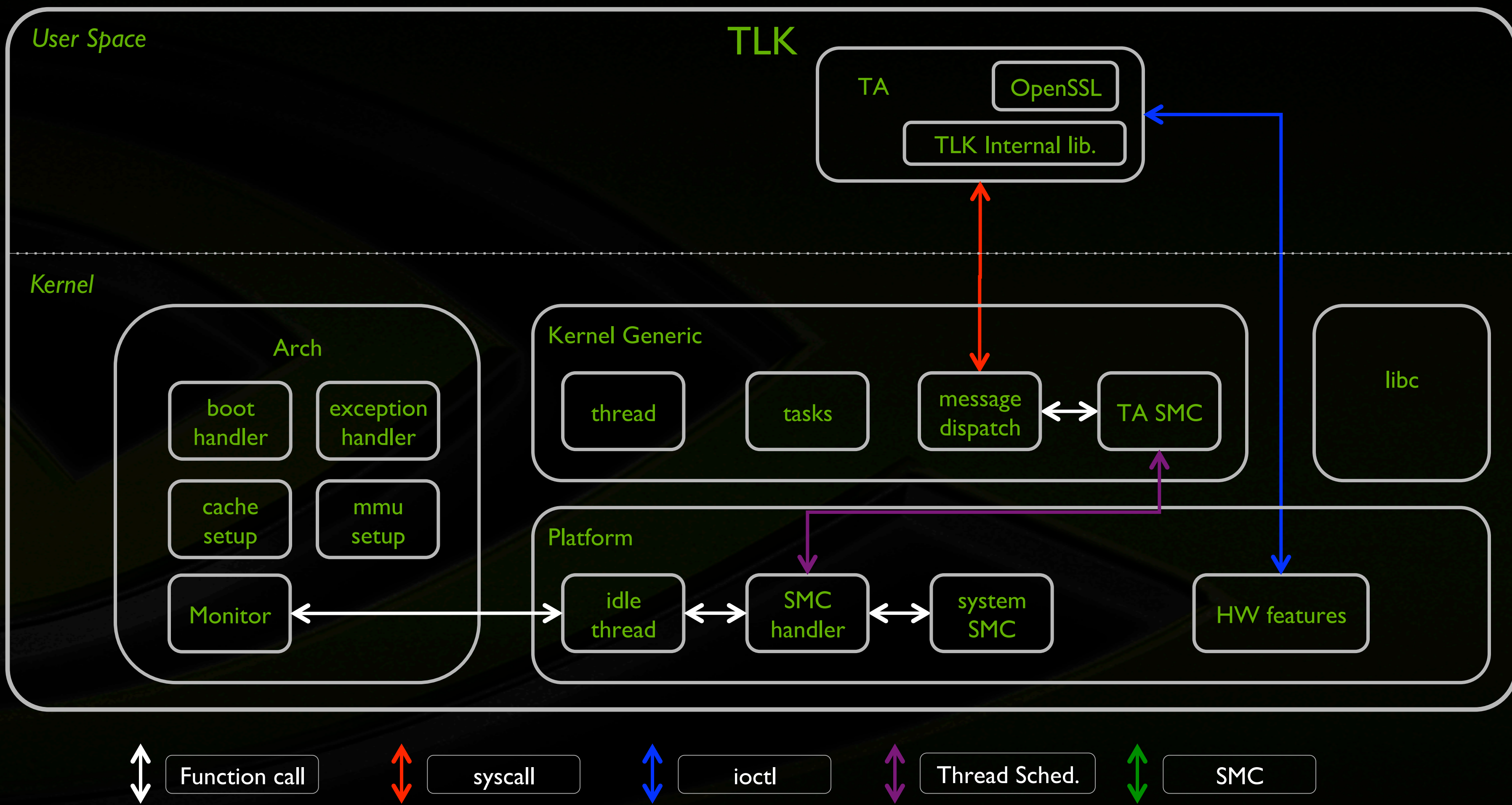
Feature Summary

- Cortex A9 & A15 support
- LP2 on slave CPU
  - Support for CPU reset after init
- Page Table Management
  - General improvements
  - Address-space separation
  - LPAE
- Addition of user mode
- 2 MB carve-out (flexible)
- Addition of syscalls
- Addition of libc
- SMC handler
- Boot to Normal World
- Many many more…

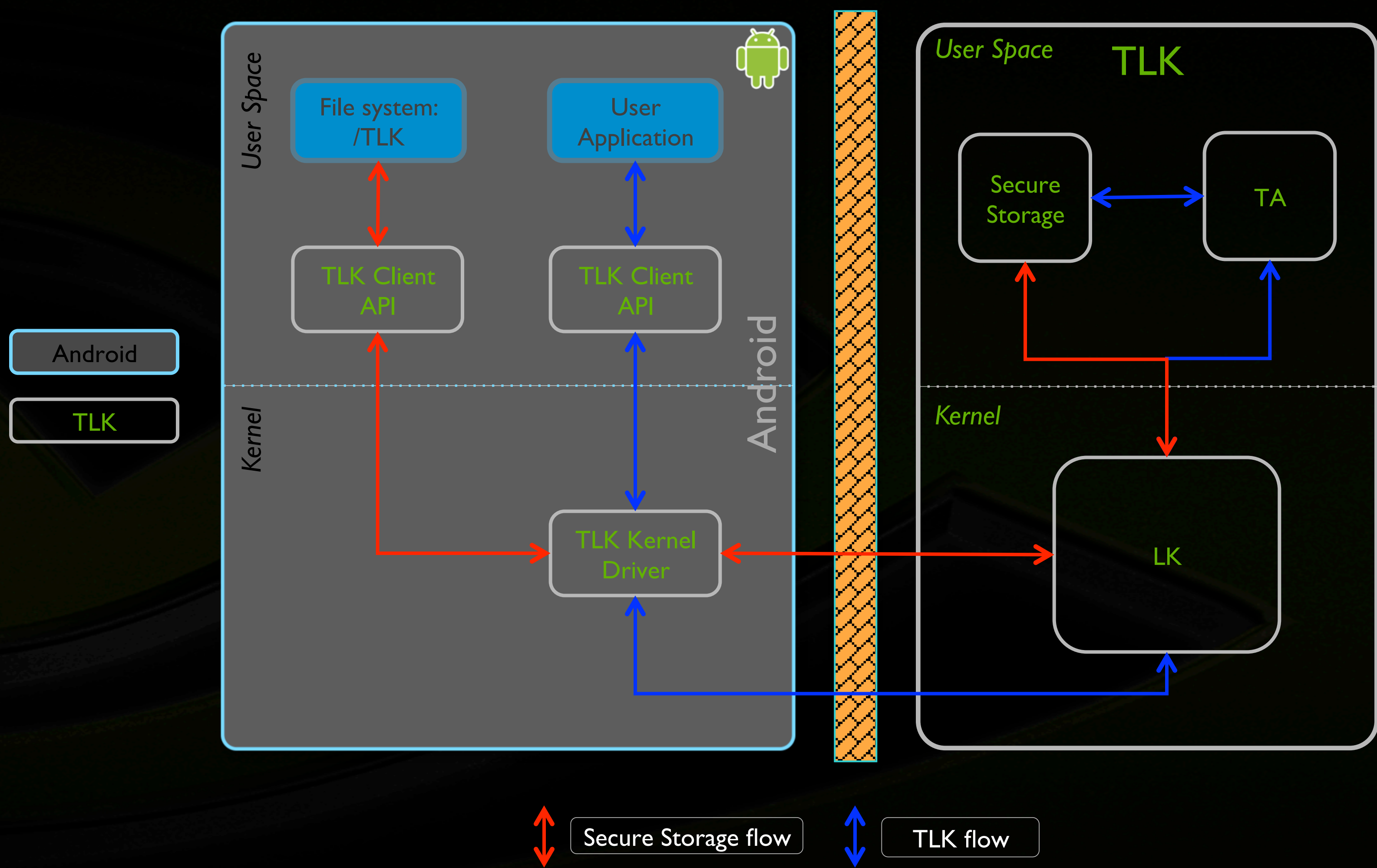
High Level Architecture



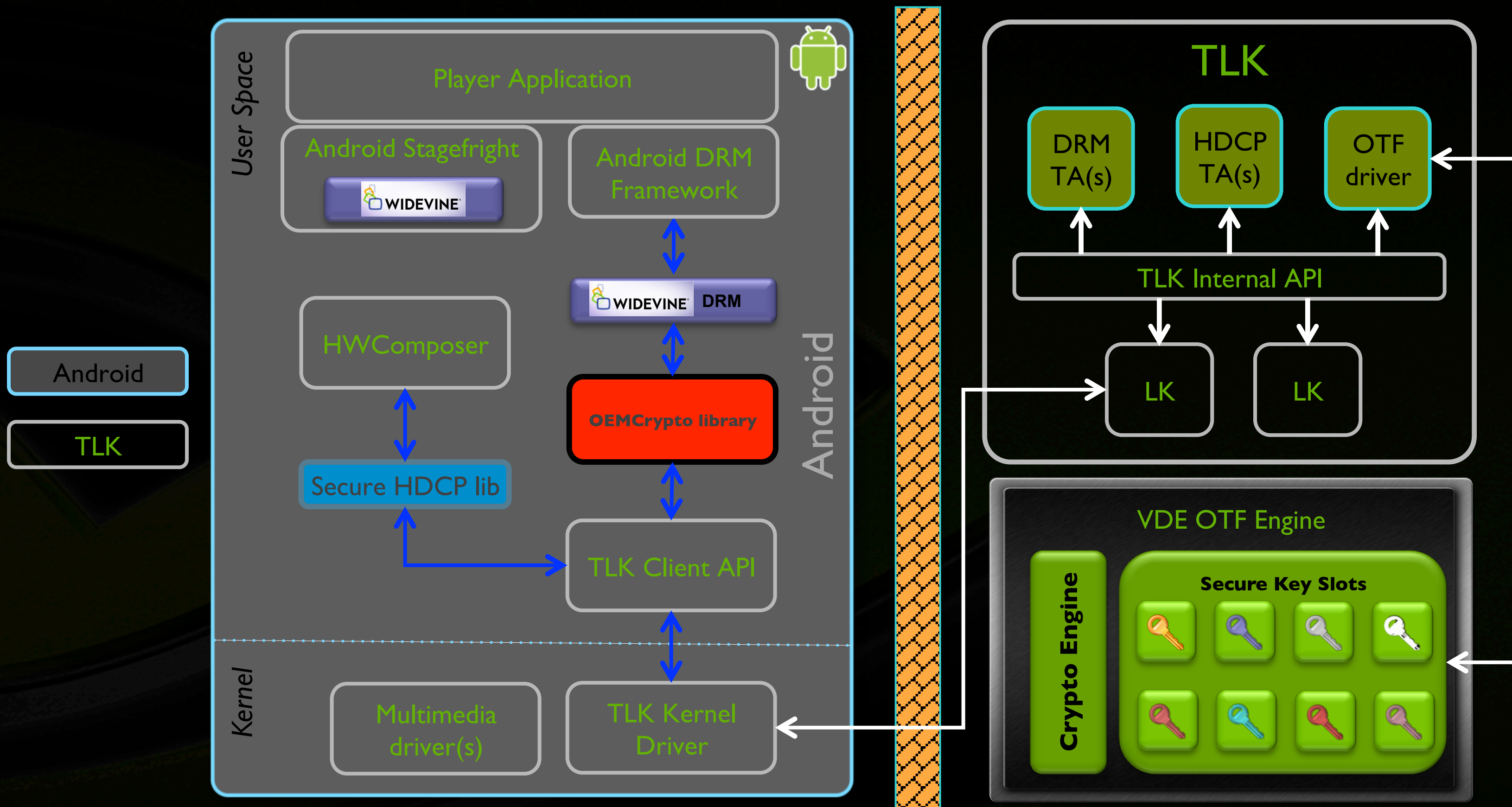
Secure World Architecture



Secure Storage



Secure (A.K.A. Protected) Content



Footprints & stats

- Memory carve-out (build-time configurable) ➪ 2MB
- TLK core code-footprint ➪ 22,843 LOC
- File count (all possible header/source) ➪ 173
- TLK library code (all possible header/source) ➪ 5,073 LOC
  - Includes Normal World client and Secure World internal libs
- Size of core tlk.bin (full support, no service) ➪ 131,072 bytes (128KB)
- Size of tlk.bin (full support, all services) ➪ 1,589,248 bytes (1.58MB)
  - secure_otf, crypto, secure_rtc, hdcp, widevine, storage (81.3% of total image)
  - Expect further savings when bionic/openssl ➪ TLK libc

Downloading TLK source

- To download TLK source code:
  - In a terminal window, set up your current working directory
  - For new trees, set up your project directory with the following shell commands:
    $ mkdir mytree
    $ cd mytree
- Download source code by entering the following shell commands
  - git clone git://nv-tegra.nvidia.com/3rdparty/ote_partner/tlk.git tlk
  - git clone git://nv-tegra.nvidia.com/3rdparty/ote_partner/lib.git lib
  - git clone git://nv-tegra.nvidia.com/tegra/ote_partner/tlk_driver.git tlk_driver
  - git clone git://nv-tegra.nvidia.com/tegra/ote_partner/tasks.git tasks
  - git clone git://nv-tegra.nvidia.com/tegra/ote_partner/daemon.git daemon

Downloading TLK source *(cont'd)*

- Directory structure
  - tlk: tlk core
  - lib: required libraries
  - tlk_driver: Linux driver between NS/S worlds
  - daemon: a proxy agent in NS world for TLK
  - tasks: containing secure task (TA)
  - tools: toolchain to build tlk

Items 3 and 4 will be released in source as example only: they will not be part of the final image

Downloading toolchains

- To download the toolchain:
  - The toolchain will not be included in the release. User needs to download toolchai
    into mytree/tools
  - Required toolchains are:
    - tools/aarch64-linux-android-4.8: for 64-bit TLK
    - tools/arm-eabi-4.7: for 32-bit TLK
- Tools could be obtained from:
  - https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/aarch64/aarch64-linux-android-4.8
  - https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.

Downloading toolchains *(cont'd)*

- Download the toolchain with the following shell commands:

```
$ mkdir mytree/tools
$ cd mytree/tools
$ git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/aarch64/aarch64-linux-android-4.8
$ git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.7
```

Building TLK image

- Directory structure
  - To make TLK image including tlk (tlk core) and lib (required libraries), run the following shell commands:

    $ cd tlk

    $ TARGET=<platform> make –e

    (#<platform> is “t124” for now)
  - The resulting binary will be at the “build-<platform>/tos.img” location
  - You can find these instructions in mytree/tlk/README file as well

Why TLK

- Designed and implemented as FOSS from day zero
  - No need for IP clean up
  - Continuous Blackduck clearance
  - Ready for secure virtualization solution
- Ready now
  - Multi-arch design from day zero
  - Productized WV and multiple PR solutions
- Scalable adoption
  - Active TLK ecosystem
  - Less than 5% SOC-specific code

Thank You

Q&A

Hadi Nahari
hnahari@nvidia.com
hadinahari
www.LinkedIn.com/in/hadinahari