Samsung

KNOX

White Paper: Samsung KNOX™ Security Solution

March 2015
Enterprise Mobility Solutions
Samsung Electronics Co., Ltd.

SAMSUNG

Contents

Section 1: BYOD and mobile security 3

Section 2: Background: What's in a smartphone? 4

- Smartphone hardware 4
- The Android operating system 5
- Boot process 6
- Mobile device security 6

Section 3: Samsung KNOX overview 7

- The Samsung KNOX philosophy 7
- Samsung KNOX design 8
- Problem: Lack of trust in Android security 11
- Solution: Base security in a hardware-rooted trusted environment 11
- Build trust 12
- Secure Boot and Trusted Boot 12
- Maintain trust 12
- Runtime protections 12
- Prove trust 13
- Problem: Mixing enterprise data and user apps on one device 13
- Solution: Protect enterprise apps and data in a secure Workspace 13
- Problem: Device theft 14
- Solution: Protect enterprise Data-at-Rest by default 14
- Problem: Difficulty of securely implementing custom enterprise applications 15
- Solution: Provide KNOX security services to enterprise applications 15
- Problem: Lack of enterprise manageability and utilities 15
- Solution: Provide extensive manageability and utilities 16

Section 4: Technology in depth 18

Part 1. Building a hardware-rooted trusted environment 19

Hardware Roots of Trust 19

Establishing trust 20

Maintaining trust 22

Proving trust 28

Part 2. Making the trusted environment enterprise ready 29

SE for Android 29

KNOX Workspace: divide and conquer 31

Sensitive Data Protection 33

On-Devce Encryption 33

Trusted Boot Based KeyStore (TIMA KeyStore) 34

Trusted Boot Based Client Certificate Management (TIMA CCM) 34

Trusted UI 34

Virtual Private Network 35

SmartCard framework 37

Single Sign-On 37

Active Directory Integration 38

Mobile Device Management 38

Simplified enrollment 40

Enterprise split billing 40

About Samsung Electronics Co., Ltd. 42

Acronyms

AES	Advanced Encryption Standard
AOSP	Android Open Source Project
BYOD	Bring Your Own Device
CAC	U.S. Common Access Card
CCM	Client Certificate Management
CESG	Communications and Electronic Security Group
CMK	Container Master Key
COPE	Corporate-Owned Personally Enabled
DAC	Discretionary Access Control
DAR	Data-at-Rest
DISA	U.S. Defense Information Systems Agency
DIT	Data-in-Transit
DoD	U.S. Department of Defense
DRK	Device Root Key
DUHK	Device-Unique Hardware Key
FIPS	Federal Information Processing Standard
IAM	Identity and Access Management
IPC	Inter Process Communication
MAC	Mandatory Access Control
MAM	Mobile Application Management
MCM	Mobile Container Management
MDM	Mobile Device Management
MMU	Memory Management Unit
NFC	Near Field Communication

Acronyms

NIST	National Institute of Standards and Technology
ODE	On-Device Encryption
OS	Operating System
PKCS	Public Key Cryptography Standards
PKM	Periodic Kernel Measurement
RKP	Real-time Kernel Protection
ROM	Read-Only Memory
RP	Rollback Prevention
SBU	Sensitive But Unclassified
SDP	Sensitive Data Protection
SEAMS	SE for Android Manager Service
SE for Android	Security Enhancements for Android
SE Linux	Security-Enhanced Linux
SRG	Security Requirements Guide
SSBK	Samsung Secure Boot Key
SSO	Single Sign-On
STIGs	Security Technical Implementation Guides
TIMA	TrustZone-based Integrity Measurement Architecture
VPN	Virtual Private Network

Section 1: BYOD and mobile security

It was only eight years ago that smartphones entered the market. Employees were already bringing their personal phones to work, but smartphones suddenly allowed access to corporate email, making it easier to respond to work-related demands after work or during business travel. Then document sharing added to the ease of doing business on mobile devices. The evolution of Bring-Your-Own-Device (BYOD) and Corporate-Owned-Personally-Enabled (COPE) started slowly, and then accelerated with the proliferation of apps for every business and personal need. While enterprise employees enjoyed the freedom and productivity of *always connected*, IT Admins were blindsided with the new and growing problem of protecting corporate intellectual property from the avalanche of unprotected personal property employees brought to work.

The security model used by IT departments was originally designed to protect an enterprise network and company-issued PCs, not the personal smartphones and tablets employees started bringing into the workplace. With both BYOD and cyber-attacks increasing, the scramble to analyze the facts and figures ensued in hopes of finding a way to manage the moving target of mobile security.

What are the numbers? What are the risks?

The Juniper Networks Mobile Threat Center, a global research facility on mobile security, released its third annual Mobile Threats Report in June 2013 from data collected from March 2012 through March 2013. They found mobile malware threats growing at a rapid rate of 614 percent to 276,259 total malicious apps, demonstrating an exponentially higher cybercriminal interest in exploiting mobile devices.[1] In another study, the 2013 Global Application Security Risk Report said 98% percent of applications presented at least one application security risk, while the average application registered 22.4 risks.[2]

Mobile malware is growing at a rapid rate of **614%**

~

276,259 malicious apps

Apps aren't the only security threat in the mobile landscape, but they are the biggest threat. A Nielsen February 2014 report, The Digital Consumer, reported that smartphone owners spend 86% of their time using apps versus the mobile web.[3] And report after report pointed to the real culprit for lack of mobile security as the Android open source code that hackers could easily use to create and distribute malicious apps.

While these facts and figures were being reported, Samsung was already at work designing solutions for the problems. In 2012, a group of Samsung engineers set out to build a trusted environment rooted in the hardware of the Android operating system (OS) that could be used cross-platform for any other OS. And, the blueprint included maintaining the trusted platform, and providing management and tools for an enterprise ready mobile security solution. In 2013, Samsung KNOX became available to enterprises large and small, giving them complete control over how they implemented their security configuration, and assurance that Samsung KNOX is a trusted mobile security solution always evolving to meet customer needs.

Section 2: Background: What's in a smartphone?

An understanding of the inner workings of an Android smartphone is helpful in appreciating the many security measures in Samsung KNOX. This section provides an overview of how Android-based phones work. Readers already familiar with Android, and mobile architectures, including ARM® TrustZone®, may wish to skip ahead to the next section.

There is much more to a smartphone than the mere apps and widgets a user typically experiences. Behind the scenes is a highly sophisticated system of advanced processor architectures, operating system kernels, libraries, and middleware and security related services. The following three sections aim to demystify each of these concepts, paving the way for a firm understanding of the security capabilities of Samsung KNOX.

Smartphone hardware

Just like a desktop computer, at the heart of every mobile device, are one or more processor cores. These are the central computational unit of the device, where all code for the phone's apps and operating system runs. The processor is also physically connected to the phone's many hardware devices. These include the antennas used for LTE, and Wi-Fi, and the internal storage drives, as well as any removable SD cards and docking ports.

One of the most important features of a processor is the use of modes of execution. A mode defines how much privilege a piece of software running on the processor has. For example, when user-installed apps run on the processor, it runs in user mode. In user mode, the apps are not allowed to directly access hardware devices or resources controlled by other apps. On the other hand, when critical operating system software is running, the processor is in privileged mode. In privileged mode, the system's software is allowed to directly access hardware devices, as well as all data held by the user's applications. Clearly, any code running in privileged mode must be protected from control by adversaries (attackers, malicious users, etc.).

KNOX leverages a processor architecture known as ARM TrustZone. While TrustZone maintains the two modes described above, it also provides a new security-specific construct called worlds. In TrustZone, there are two worlds, the normal world, and the Secure World. Virtually all smartphone software as we know it today still runs in the normal world. The Secure World is reserved for highly-sensitive computations such as those involving cryptographic keys (see the Mobile Security section below). As described throughout this document, KNOX makes extensive use of TrustZone's Secure World, both for protecting enterprise confidential data, and for monitoring the OS kernel running in the normal world. Given these highlights of the TrustZone processor architecture, the next section explains two more security critical components, the Android OS, and its kernel.

The Android operating system

In this section, we examine the basic structure of the Android OS, which KNOX is built on. Recall that a hardware processor provides two modes, user and privileged. Operating systems use both of these modes for various functions. The portion running in privileged mode is called the kernel. OS kernels are among the most rigorously engineered pieces of software in the world, because they must perform many functions, all with the power of the processor's privileged mode. For example, any time data arrives for the phone from the Internet, the OS kernel first chooses whether to even allow the data to proceed, or to drop it if it seems unwanted. If the data is allowed, the kernel examines it and decides which application on the phone the data is intended for. The kernel then places the data in the app's memory, and notifies the app that data has arrived. If the app then wishes to send a reply, the app's reply is sent by repeating this whole process in reverse.

Given this example, consider what could happen if an attacker gained control of the OS kernel. Due to the kernel's high permissions, the attacker could tamper with and leak arbitrary sensitive data from any application, and send it to anywhere on the Internet. This is why KNOX implements the extensive protections for OS kernels covered in later sections.

In most traditional operating systems, when applications wish to communicate with each other, they ask the kernel to set up the lines of communication for them. To facilitate more rich and easy to use forms of app communication, the Android OS instead provides another layer of software, fittingly called the middleware.

The Android middleware runs in user mode, but sits between the kernel and apps. The middleware provides a rich set of communication methods that allow apps to share their data with each other and perform operations on each other's behalf. For example, many image library apps offer the option to take photos, even though they don't know how to use the phone's camera. Instead, they simply request that an app that does understand the camera take the picture on their behalf. A second major function of KNOX is to ensure that such forms of communication do not occur between enterprise apps and user apps. This prevents sensitive data from being leaked to an untrusted third-party, and prevents corrupted data from entering enterprise apps.

Boot process

An important concept that ties together the hardware, kernel, and apps is the boot process. When a device is first turned on, the user's applications are not immediately available. Instead, a chain of software components start, with each component starting the next one in the chain. Typically, when the user presses the ON button, the device first runs a program called a boot loader. Many mobile device architectures use multiple levels of bootloaders to perform different functions. The boot loader then finds where the kernel is stored, and begins running the kernel in the processor's privileged mode. The kernel starts the Android middleware and some basic apps, running them in user mode. At this point, the user is presented with a login screen, and can begin using the phone.

Mobile device security

This final section explains some important concepts in the security of mobile devices. An significant detail that differentiates mobile security from other domains is that device owners have complete control over how to use their own devices, as opposed to, say, a corporate-owned laptop, which is controlled by IT administrators. For example, in a BYOD scenario, sensitive emails are likely be downloaded to the device, but the user may simultaneously compromise the kernel's security to allow for their own device customizations. This process is typically known as device rooting.

Even though users' motivations for rooting are often benign, such as installing custom themes, the subsequent security breach makes it easier for malicious parties to gain control of the device. The same techniques have been known to be used by malware authors as such access can steal the user's data or use their device to attack others. Many of the security measures put in place by KNOX are designed to either prevent device rooting, or to mitigate the resulting damage.

Another fundamental feature for mobile device security is the use of cryptography. KNOX uses cryptography for three key functions:

- **Encryption** - the scrambling of data using a protected key to keep it confidential
- **Hashing** - the creation of a unique series of numbers to represent a particular piece of software or data; a single difference in a piece of data yields a different hash.
- **Signing** - the encryption of a hash of a piece of data using a private key to prove that the data originated from a particular party

KNOX frequently uses signing to produce signatures of hashes of firmware components. This proves that the firmware component originated from the owner of the private key used for signing, and in this case, it proves the component originated from Samsung. KNOX maintains signing keys are only accessible in the TrustZone Secure World.

Section 3: Samsung KNOX overview

Enterprise data is increasingly finding its way onto smartphones as a result of BYOD and COPE policies.

This new way of working has increased productivity for employees by placing work and personal data, such as emails, on the same device. However, this has also greatly complicated the task of IT security. Mobile devices provide numerous avenues through which sensitive data can fall into the wrong hands, such as sharing of data with untrusted third-party applications, device theft, intentional rooting by power users, and misconfigured or vulnerable enterprise applications. These problems are manifest across organizations with hundreds or thousands of devices, all requiring security management and configuration.

In this whitepaper, we present Samsung KNOX. KNOX aims to be the most comprehensively secure and manageable mobile device solution for enterprises large and small. Based on the Android OS, Samsung KNOX is designed around the philosophy that the foundations of device security should be rooted in fixed hardware mechanisms. KNOX bases this foundation in the principles of trusted computing, a set of methods for making devices that can prove to enterprises they are running the correct security software, and can raise alerts in the event that tampering is detected. On top of this trusted foundation, KNOX builds a Workspace environment to protect enterprise apps and their data, a robust set of data at rest protections, and a large suite of enterprise security tools, including a highly configurable Virtual Private Network (VPN) and Mobile Device Management (MDM) interfaces.

We begin our overview with the design philosophy that is behind everything we have built in KNOX.

The Samsung KNOX philosophy

KNOX is built using a two-step design philosophy:

Step 1. Build a trusted environment rooted in hardware security mechanisms.
In a trusted environment, sensitive or enterprise-critical functionality is only enabled once the device is in an allowed state. Here, state refers to the security-relevant software and configurations on the device. For example, parts of state considered by KNOX include the bootloaders, kernel, and TrustZone OS, as well as security policy configurations. Furthermore, the trusted environment allows for remote attestation, where any change can be securely summarized via a set of proofs. Third parties can then inspect these proofs to decide if the device state meets their security requirements. A trusted environment is considered hardware rooted if its security is based on the high difficulty of physically tampering with hardware circuits.

Why is it important to root trust in hardware? The designers of KNOX are aware that throughout the history of operating system security, certain critical functionalities have always been trusted to privileged system software (mainly the OS kernel). However, in the last two decades, attackers have become more and more successful at exploiting kernel flaws whether on their own or others' devices. Other mechanisms such as heavyweight virtual machines or special Basic Input/Output System (BIOS) checks have been implemented and circumvented. The KNOX design recognizes that to date, the single best defense against full-system compromise is to tie any system self-checks to a secret maintained by secure hardware, which is out of the reach of any software-based adversary, and virtually all physically present adversaries as well.

Step 2. Make the trusted platform ready for enterprise use.
The trusted platform must be usable by enterprises. This involves giving enterprises complete control and configurability over their data. The KNOX Workspace protects enterprise data using encryption, and the enterprise manages the Workspace using Mobile Device Management (MDM) capability. In addition, KNOX supplies a collection of useful secure applications and utilities, such as VPN, that allow enterprise-ready deployment. The security of the KNOX Workspace and utilities is firmly grounded in the hardware root of trust described in the first step, and in isolating the Workspace from the personal space.

Samsung KNOX design

Through this two-step design philosophy, KNOX addresses the most pressing security problems facing enterprises' BYOD and COPE strategies today. To this end, we identify the following key challenges in making an Android-based system enterprise ready:

1. The problem of device rooting
2. Mixing enterprise data with user apps on the same device
3. Device theft
4. Difficulty of securely implementing custom enterprise applications
5. Lack of enterprise manageability and utilities

The following sections detail each of these problems, followed by the solution stemming from the KNOX design philosophy.

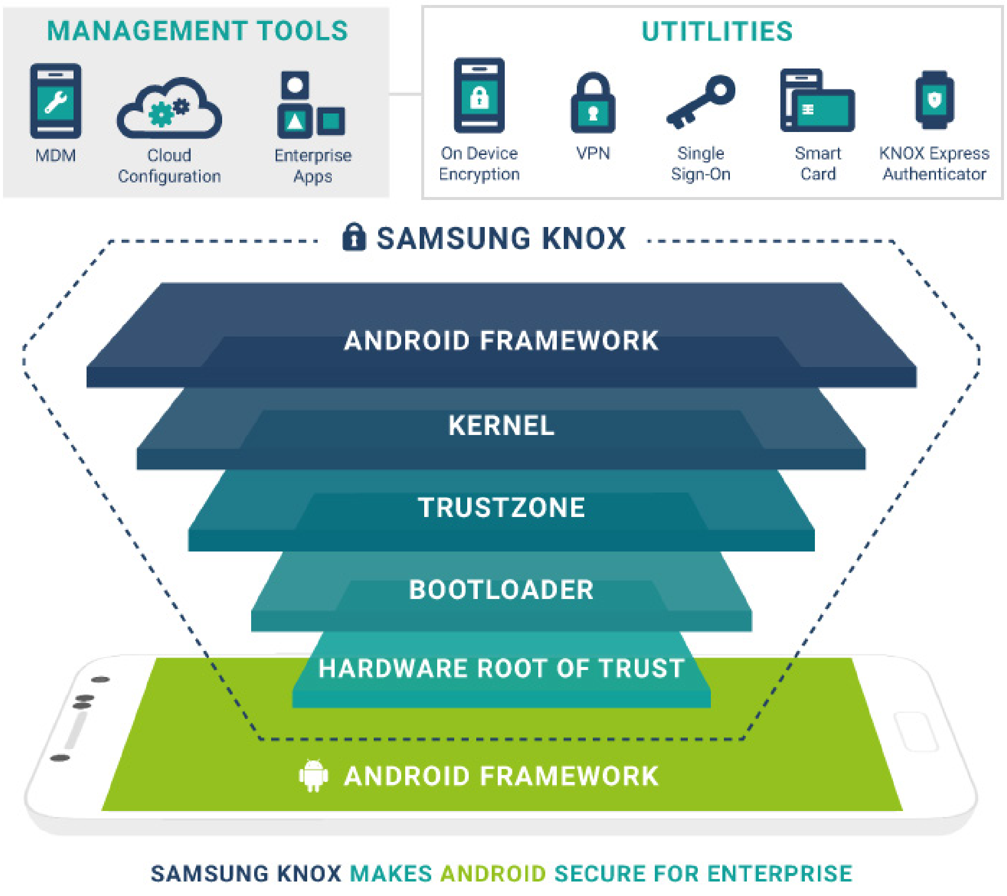

Figure 1 – Samsung KNOX Enterprise Security Solution

Table 1 summarizes the structure of the following section by pairing each of the above listed hurdles to enterprise adoption of Android with the KNOX-specific solution.

KNOX Strategy	Problem(s) Solved	Solution Technologies
Build a Hardware-Rooted Trusted Environment	Lack of trust in Android security	**Hardware Root of Trust** Samsung Secure Boot Key, Rollback Prevention Fuses, KNOX Warranty Bit, Device Root Key (DRK) **Build Trust** Trusted Boot using TrustZone-Based Integrity Measurement Architecture (TIMA), Rollback Prevention **Maintain Trust** Real-Time Kernel Protection (RKP), Periodic Kernel Measurement (PKM), DM-Verity **Prove Trust** TIMA Attestation
Make Trusted Environment Enterprise-Ready	1. Mixing enterprise data and user apps on one device	1. KNOX Workspace, Security Enhancements for Android
	2. Device theft	2. KNOX Workspace Encryption, Sensitive Data Protection (SDP), On-Device Encryption (ODE)
	3. Difficulty of securely implementing custom enterprise applications	3. TIMA KeyStore, Client Certificate Manager (CCM), SE for Android Management Service (SEAMS)
	4. Lack of enterprise manageability and utilities	4. Mobile Device Management (MDM), Virtual Private Network (VPN), Active Directory Integration, Single Sign-On (SSO)

Table 1 - KNOX Solution Technologies

Problem: Lack of trust in Android security

The Android OS was originally designed for end users and not for enterprise use. The original Android approach to security was to simply isolate apps from each other. However, this alone is insufficient to provide confidence for enterprise use. For example, how can enterprises be confident that security measures are even enabled, when it is common practice for users to root their devices? (Device rooting is the practice of intentionally exploiting privileged software to circumvent vendor-included restrictions.)

Solution: Base security in a hardware-rooted trusted environment

KNOX provides strong guarantees for the protection of enterprise data by building a hardware-rooted *trusted environment*. A trusted environment ensures that enterprise-critical operations, such as decryption of enterprise data, can only occur when the device is proven to be in an allowed state. For many pieces of device software, such as the kernel and TrustZone apps, the *allowed* state is represented by the cryptographic signature of each piece of software. A trusted environment is hardware-rooted if both the cryptographic keys and code used to compute these signatures are tied back to unmodifiable values stored in hardware. KNOX facilitates a hardware-rooted trusted environment in three steps.

- Build trust by enforcing that only approved versions of system-critical software are loaded
- Maintain trust by ensuring that system-critical software is not modified once loaded
- Prove that only approved system-critical software is loaded and run on a particular device when requested to do so by the enterprise.

Figure 2 below shows how KNOX builds, maintains, and attests its trusted environment.

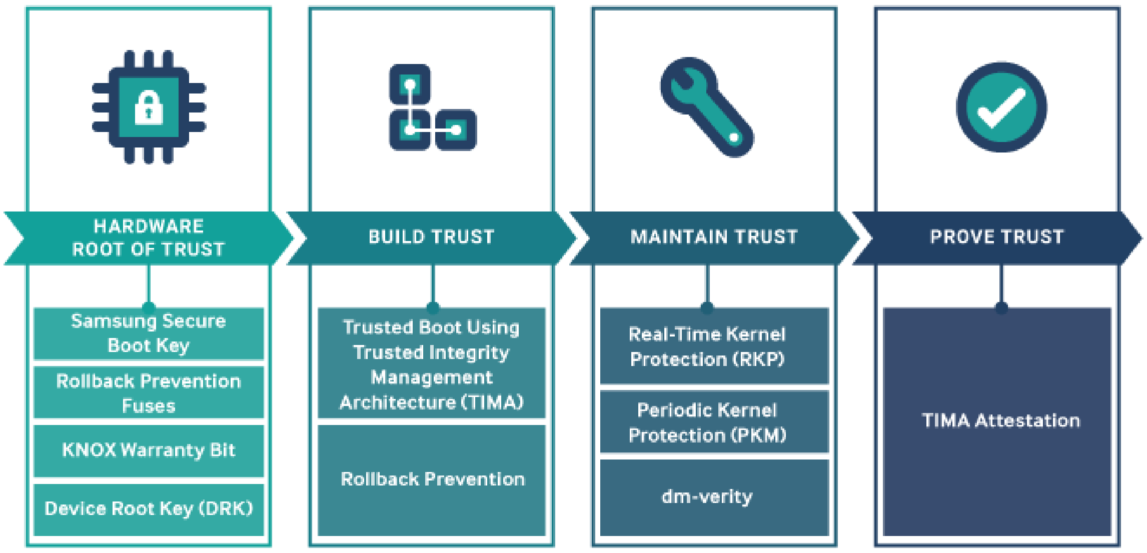

Figure 2 - KNOX Builds, Maintains and Attests Its Trusted Environment

Build trust

Secure Boot and Trusted Boot

When a device is first turned on, the user’s applications are not immediately available. Instead, a chain of software components is started, with each component starting the next one in the chain. Typically, once the hardware is powered on, it first runs a program called a bootloader, which in turn runs the operating system’s kernel, a highly privileged component that starts applications and can access storage and network devices directly.

Many device vendors support a process known as Secure Boot, and Samsung KNOX devices are no exception. In a Secure Boot process, each component in the boot chain (bootloader, kernel, etc.) checks the integrity of the next component through signature verification. If the signature verification fails, the boot process is stopped.

Secure Boot is limited because it cannot distinguish between different approved versions, for example, a bootloader with a known vulnerability and a later patched version, since both versions have valid signatures. To address this limitation, Samsung KNOX adopts Trusted Boot in addition to Secure Boot. In the Trusted Boot process, each software component in the chain measures and securely stores the cryptographic hash of the next component in TrustZone Secure World memory before loading it. Storing these measurements allows a third-party to identify the exact versions of software loaded on the device through the process of attestation. For example, this can be used to verify that only the latest patched versions of software are run, complementing the Rollback Prevention feature that ensures patched software is not downgraded to a vulnerable version.

If signature verification fails, KNOX either records the tampering by blowing a one-time fuse, called the KNOX warranty fuse, or by preventing further booting, depending on the configuration. Devices that have the fuse set cannot run certain KNOX features such as the KNOX Workspace thereafter.

Both Secure Boot and Trusted Boot have their trust rooted in hardware. The first piece of software loaded is the primary boot loader, which is kept in hardware-protected Read-Only Memory (ROM). In addition, the cryptographic key used to verify signatures is the Samsung Secure Boot Key, also stored in hardware fuses.

Maintain trust

Runtime protections

At the end of a successful secure boot, only approved versions of system software (such as the TrustZone OS) have been loaded. However, they can then be modified. For example, users may either intentionally or unintentionally run code that exploits a flaw to maliciously modify the kernel, thus bringing it under adversarial control. KNOX detects kernel compromises quickly using a pair of techniques: Real-time Kernel Protection (RKP) to actively prevent kernel code modification, and Periodic Kernel Measurements (PKM) that periodically check kernel code integrity. Both RKP and PKM checks are performed from the TrustZone Secure World.

The kernel is not the only attractive target for malware and malicious users. There are large numbers of other code objects and configurations that can be used by malware to become persistent, meaning that it can restart itself each time the device restarts. KNOX prevents such modifications by integrating Google’s DM-Verity, a kernel module that verifies the integrity of applications and data stored on the critical system partition. In the event a malicious process or user modifies something on the system partition, DM-Verity detects the modification the next time the data is read, and blocks any attempted access to modified data.

Prove trust

Consider an MDM server that wishes to interact with a mobile device. The MDM should not simply assume that the device is uncompromised. Instead, a KNOX-enabled device provides the MDM with an attestation, a cryptographically verifiable collection of device state measurements. This includes hashes of the bootloaders, kernel, TrustZone OS, and logs from runtime protection mechanisms, among others. (For the complete contents of the attestation message, see TIMA Attestation in Section 4: Technology in depth.) The MDM can then decide if all entries on the list are approved. The attestation is signed using a key derived from the Device Root Key (DRK), which is hardware protected. Thus, if we trust the hardware to be untampered and the ARM TrustZone Secure World software to work properly, a trusted environment is established, and this can be proved to a third party.

The problems in subsequent sections are solved using technologies built on top of this trusted environment. We stress that without the trusted environment covered in this section, all remaining security measures are ineffective, as there is no guarantee that they were even loaded and run as expected.

Problem: Mixing enterprise data and user apps on one device

With the emergence of BYOD and COPE, one of the major challenges facing enterprises is the mixing and interaction of sensitive enterprise apps and data with potentially malicious user-installed apps on the same device. Android provides apps with many ways to interact with one another. Apps may share databases containing photos or contact information, and perform actions on each other's behalf, such as a browser opening a link in an SMS text message. This design has greatly benefited the mobile ecosystem, resulting in an explosion of useful applications. However, this ease of sharing can be problematic where data from sensitive enterprise emails and documents can easily be leaked to untrusted apps that claim to provide a necessary functionality. Enterprises must have solid guarantees that their data is safe even with hundreds or thousands of employees downloading untrustworthy third-party apps.

Solution: Protect enterprise apps and data in a secure Workspace

To solve this problem, we needed to provide strong isolation between the enterprise and user aspects of the device. Such strong isolation guarantees are provided by Mandatory Access Controls (MAC). In a MAC system, access to resources is restricted using a policy that can only be modified by the device vendor, in this case, Samsung.

Therefore, we decided to adapt the Security Enhancements for Linux (SELinux) MAC system for KNOX. SELinux provides a rich policy language for describing fine-grained access to resources by programs. We extend SELinux into Security Enhancements for Android (SE for Android). SE for Android provides additional mediation locations in KNOX's Android middleware, along with additional policy language. KNOX's pioneering of SE for Android has now led to its adoption into the Android Open Source Project (AOSP).

The KNOX Workspace is built on top of SE for Android to define a protected environment for enterprise data and apps. The Workspace provides a full environment, including the home screen, launcher, applications, and widgets. This environment runs alongside the user's environment, but it is protected from interference from user-installed applications. All data created by container applications is kept on a protected partition as described in the following section. In the event that tampering is detected during Trusted Boot, the container and its data are no longer accessible.

Problem: Device theft

Smart phone theft is one of the most serious threats to the confidentiality of enterprise data. A 2014 study, "Smart phone thefts rose to 3.1 million last year, Consumer Reports finds," estimated that only 7% of smartphone users enable encryption for Data at Rest. Furthermore, only 36% enable a screen lock, and 34% take no security precautions at all.[4] Also, a recent FCC study cites FBI data estimating that nearly 10% of all thefts and robberies in the US in 2013 were related to theft of a mobile device.[5] Given the high prevalence of mobile device theft, and the reluctance of users to secure their data in the event of theft, a secure mobile OS must protect data without user initiative.

Solution: Protect enterprise Data-at-Rest by default

Samsung KNOX does not depend on users to secure their own BYOD devices in case of loss or theft. KNOX defines two classes of data – *protected* and *sensitive*. All data written by apps in the secure Workspace is considered protected. Protected data is encrypted on disk when the device is powered off. In addition, the decryption key for protected data is tied to the device hardware. This makes protected data recoverable only on the same device. Furthermore, access controls are used to prevent applications outside the KNOX Workspace from attempting to access protected data.

Even stronger protection is applied to *sensitive* data. Sensitive data remains encrypted as long as the Workspace is locked, even if the device is powered on. When the user unlocks their KNOX Workspace using their password, Sensitive Data Protection (SDP) allows sensitive data to be decrypted. When the user re-locks the Workspace, SDP keys are cleared. The SDP data decryption key is tied to both device hardware and to the user input. Therefore, the data is recoverable only on the same device and with user input.

SDP can be used in one of two ways. First, all emails received are considered sensitive, and are immediately protected by SDP encryption. Emails received when the Workspace is locked, are immediately encrypted, and can only be decrypted the next time the Workspace is unlocked. The second way to use SDP is through the KNOX Chamber. The Chamber is a designated directory on the file system. Any data placed into the Chamber is automatically marked as sensitive and protected by SDP.

Problem: Difficulty of securely implementing custom enterprise applications

Properly implementing cryptographic, authentication, and secure storage services has traditionally been challenging. Vulnerabilities are regularly found in both cryptographic libraries and applications that leak keys. Once a key is leaked, all data previously encrypted with that key becomes vulnerable. Keeping secret keys secret is a problem for a number of reasons. First, many applications make multiple copies of keys in their internal logic, which they do not properly track and delete, thus increasing the risk of key leakage. Aggravating the problem, implementation flaws, such as the now infamous Heartbleed bug, can allow secret keys to be leaked directly to the network. In spite of the risks associated with implementing cryptographic services in applications, many enterprise apps require them.

Solution: Provide KNOX security services to enterprise applications

KNOX allows enterprise developers to build custom applications on top of its hardware-rooted trusted environment. KNOX exposes APIs for key management, and FIPS 140-2 compliant cryptographic algorithms. The Trusted Boot-based TIMA KeyStore provides applications one type of secure key storage. Recall that Trusted Boot only allows sensitive operations to occur if approved versions of all security-critical system software are loaded. The TIMA KeyStore stores all application keys in the TrustZone Secure World storage. From there, the keys can only be accessed if Trusted Boot is successful. Thus, an application's keys are safe, even in the event that a user or malicious app tampers with critical system software.

Similar to TIMA KeyStore, Client Certificate Management (CCM) is a complementary service for generating, storing and using asymmetric key pairs and certificates in the TrustZone Secure World storage. The CCM API provides applications with PKCS#11 compliant token management, as well as public key algorithms for signatures and encryption.

Problem: Lack of enterprise manageability and utilities

KNOX introduced the functionality and manageability required for enterprise use. First, to use the secure environment effectively, enterprises need utilities such as VPN and Microsoft Exchange integration. Second, enterprises need complete control to configure the Workspace to meet their security needs. We realized that each enterprise balances security and functionality differently. For example, enterprises have widely differing password policies. Further, enterprises have lists of approved applications, identity providers, and IT partners they trust to deliver their IT infrastructure.

Solution: Provide extensive manageability and utilities

Enterprises large and small have very diverse sets of needs when it comes to device management. Samsung KNOX gives enterprises complete control to configure the Workspace to their needs using an extensive set of more than 1500 Mobile Device Management (MDM) APIs.

Second, Samsung KNOX provides utilities that allow ready deployment in enterprises. Samsung KNOX offers per-application VPN controls, a smartcard framework, and Single Sign-On (SSO) integration with Microsoft Active Directory. These features enable Samsung KNOX to easily integrate into any enterprise.

MDMs can obtain proof that their devices are running a trusted environment using TIMA attestations. The TIMA attestation data contains the cryptographic hashes of the boot components and other critical security information such as if SE for Android is enabled. This data is signed using a key derived from the DRK, which proves that the attestation data originated from the TrustZone Secure World on a particular Samsung device.

For large enterprises requiring extensive customization above what is offered by MDMs, Samsung KNOX has a dedicated team to determine unique needs and prepare custom KNOX flavors.

Finally, to build enterprise confidence, Samsung KNOX has obtained a number of certifications:

FIPS 140-2 Certification	Issued by the National Institute of Standards and Technology (NIST), the Federal Information Processing Standard (FIPS) is a US security standard that helps ensure companies that collect, store, transfer, share, and disseminate sensitive but unclassified (SBU) information and controlled unclassified information (CUI) can make informed purchasing decisions when choosing devices to use in their workplace. Samsung KNOX meets the requirements for FIPS 140-2 Level 1 certification for both data-at-rest (DAR) and data-in-transit (DIT).
DISA Approved STIG	The Defense Information Systems Agency (DISA) is an agency within the US DoD that publishes Security Technical Implementation Guides (STIGs) which document security policies, requirements, and implementation details for compliance with DoD policy. On April 17, 2014 DISA approved the STIG for Samsung KNOX 1.0.
DISA Approved Product List	On May 14, 2014 DISA added five KNOX-enabled devices to the US DoD Approved Products List (APL). ***NOTE**: The five Samsung devices added to DISA's APL are the only Android 4.4 OS devices on the list as of May 14, 2014. They are also the only devices certified under Common Criteria (CC) by Mobile Device Fundamental Protection Profile (MDFPP). These five new devices represent a twenty percent increase of mobile products available for purchase in the DoD.*
Common Criteria Certification	The Common Criteria for Information Technology Security Evaluation, commonly referred to as Common Criteria, is an internationally-recognized standard for defining security objectives of information technology products and for evaluating vendor compliance with these objectives. A number of Governments use Common Criteria as the basis for their own certification schemes. Select Galaxy devices with KNOX embedded received Common Criteria (CC) certification on February 27, 2014. The current CC certification targets the new Mobile Device Fundamentals Protection Profile (MDFPP) of the National Information Assurance Partnership (NIAP), published in October 2013, which addresses the security requirements of mobile devices for use in enterprise.
CESG Approved	The Communications and Electronic Security Group (CESG) approved KNOX-enabled Android devices for United Kingdom government use on May 14, 2014.
FICORA	FICORA: Samsung devices with KNOX fulfill national security requirements as defined by the Finnish National Security Auditing Criteria (KATAKRI II).
ASD	Australian Signals Directorate: ASD endorsing the Protection Profile for Mobile Device Fundamentals as well as recognizing evaluations against this Protection Profile.
NIAP-validated	Samsung KNOX is approved by the United States government as the first NIAP-validated consumer mobile devices to handle the full range of classified information.

For the most up-to-date list of certifications, please visit https://www.samsungknox.com/en/security-certifications .

Section 4: Technology in depth

The KNOX design philosophy consists of two steps:

1. Build a hardware-rooted trusted environment. A trusted environment ensures that sensitive and enterprise-critical operations only occur once the device is proven to be in an allowed state. Part 1 examines how KNOX builds a hardware-rooted trusted environment and how it can prove to a third-party that it runs a trusted environment.

2. Make the trusted environment enterprise ready. KNOX provides enterprise security services such as VPN, secure storage, cryptographic APIs and secure isolation of enterprise apps from untrustworthy user apps, just to name a few. Part 2 delves into each of these, relating each back to the trusted environment provided in Part 1.

Subsequent sections detail the two steps. Figure 3 is an overview of the KNOX design.

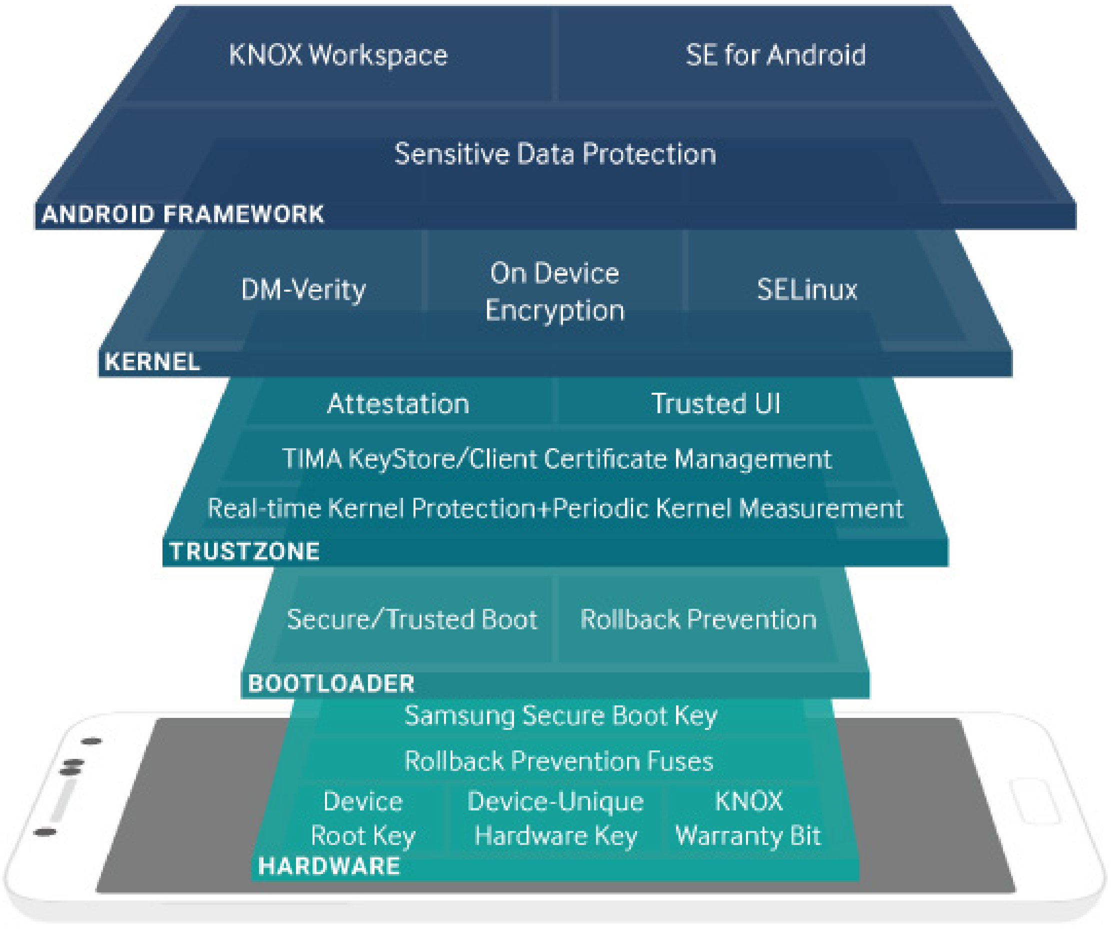

Figure 3 - KNOX Architecture Overview

Part 1. Building a hardware-rooted trusted environment

This first part examines how KNOX builds its trusted environment in four subsections. First, we examine the hardware roots of trust, which trust in all other components relies upon. Second, we present how KNOX establishes trust during boot time. Third, we show how KNOX maintains the already established trust while the system is in use. Finally, we examine how KNOX proves its trustworthiness to remote parties such as the enterprise management system.

Hardware Roots of Trust

In this section, we describe the hardware components that are the foundation of Samsung KNOX's trusted environment.

Device-Unique Hardware Key (DUHK). The DUHK is a device-unique symmetric key that is set in hardware at manufacture time in the Samsung factory. The DUHK provides a way to bind data to a particular device as follows. The DUHK is only accessible to a hardware cryptography module and is not directly exposed to any software. However, software can request for data to be encrypted and decrypted by the DUHK. Data encrypted by the DUHK becomes bound to the device, because it cannot be decrypted on any other device. The DUHK is typically used to encrypt other cryptographic keys.

Samsung Secure Boot Key (SSBK). The SSBK is an asymmetric key pair used to sign Samsung-approved executables of boot components. The private part the SSBK is used by Samsung to sign the secondary and application bootloaders. The public part of the SSBK is stored in hardware one-time programmable fuses at manufacture time in the Samsung factory. The Secure Boot process uses this public key to verify whether each boot component it loads is approved (See the section on Secure Boot).

Rollback Prevention Fuses (RP Fuses). The RP fuses are hardware fuses that encode the minimum acceptable version of Samsung-approved bootloaders. Old software may contain known vulnerabilities that can be exploited. The rollback prevention feature prevents approved, but out-of-date versions of bootloaders from being loaded (See the section on Rollback Prevention). The version number in the RP fuses is set when system software is first installed and later during updates. RP fuses are one-time programmable. Thus, the minimum acceptable version can only be incremented but not decremented.

KNOX Warranty Fuse. The KNOX warranty bit is a one-time programmable fuse that signifies whether the device has ever been booted into an unapproved state. If the Trusted Boot process detects that non-approved components are used, or if certain critical security features such as SELinux are disabled, it sets the fuse. Thereafter, the device can never run Samsung KNOX, access to the DUHK and DRK in the TrustZone Secure World is revoked, and the enterprise's data on the device cannot be recovered.

ARM TrustZone Secure World. The Secure World is a hardware-isolated environment in which highly sensitive software executes. The ARM TrustZone hardware enforces that memory and devices that are marked secure can only be accessed in the Secure World. Most of the system as we know it, including the kernel and middleware, as well as all apps, execute in what is called the normal world. Normal world software can never access the data used by Secure World software. The Secure World software, on the other hand, is more privileged, and can access both secure and normal world resources. KNOX makes extensive use of the Secure World both for cryptographic operations, and for monitoring normal world security.

Bootloader ROM. The primary bootloader (PBL) is the first piece of code to be run during the boot process. The PBL is trusted to start the measurement and verification of the boot chain (see sections on Secure Boot and TIMA Trusted Boot). To prevent tampering, the PBL is kept in secure hardware Read Only Memory (ROM). The device hardware loads and runs the PBL from ROM at boot, and the PBL starts the Secure and Trusted Boot processes.

Device Root Key (DRK). The DRK is a device-unique asymmetric key pair that is signed by Samsung's root key through an X.509 certificate. This certificate proves that the DRK was produced by Samsung. The DRK is generated at manufacture time in the Samsung factory and is stored on the device encrypted by the DUHK, thus binding it to the device. The DRK is only accessible from within the TrustZone Secure World.

Because the DRK is device-unique, it can be used to tie data to a device through cryptographic signatures. The DRK is not used directly to sign data; instead, signing keys are derived from the DRK. As an example, the TIMA attestation data, which proves the device is in a trusted state, is signed using the Attestation Key, which is itself signed by the DRK. The DRK signature proves that the attestation data originated from the TrustZone Secure World on a Samsung device. Note that while the DRK is not stored directly in hardware, it is an important part of the root of trust as it derives other signing keys, and is protected by both the DUHK and TrustZone Secure World.

Establishing trust

Android begins the startup process with the primary bootloader, which is loaded from ROM. This code performs basic system initialization and then loads another bootloader, called a secondary bootloader, from the file system into RAM and executes it. Multiple secondary bootloaders may be present, each for a specific task. The boot process is sequential in nature with each secondary bootloader completing its task and executing the next secondary bootloader in the sequence, finally loading the application bootloader known as aboot, which loads the Android operating system. This sequence of components is called the boot chain.

Secure Boot

In the Secure Boot process, each component in the boot chain verifies the integrity of the subsequent component against a signature before executing it. If verification fails, the boot process is stopped. Signatures of boot components are generated at build time using the Samsung Secure Boot Key (SSBK). The public part of the SSBK is stored in hardware fuses during manufacture. The first component in the chain, the primary bootloader, is stored in immutable ROM and is trusted to verify the secondary bootloader. Thus, the Secure Boot chain can only be compromised by hardware tampering. Later boot components, such as the kernel, are signed by another Secure Boot Key that is programmed into the previous boot component.

TrustZone-based Integrity Measurement Architecture

Secure Boot prevents the device from starting if unapproved boot components are detected. However, if the device does start, Secure Boot cannot inform a third party about what versions of approved boot components have been loaded and run. For example, it cannot distinguish between a boot component with a known vulnerability vs. a later patched version, since both versions have valid signatures. In addition, some carriers decide to allow custom OS kernels to run on their devices. On these devices, Secure Boot cannot prevent unapproved kernels from running. This clearly poses a threat to enterprise applications and data. To improve upon this limitation of Secure Boot, KNOX contains the TrustZone-based Integrity Measurement Architecture (TIMA). TIMA introduces two features: Trusted Boot and Attestation.

TIMA Trusted Boot

In Trusted Boot, each boot component in the boot chain measures the subsequent component and stores the measurement before executing it. An illustration of Trusted Boot is given in Figure 4. The measurement is a SHA256 cryptographic hash of the boot component. These hashes are securely stored in TrustZone-protected memory. The set of hashes consists of one or more secondary bootloaders, the TrustZone Secure World operating system, the application bootloader and the normal world kernel. Also, depending on the processor make and model, hashes of additional firmware images such as the modem are included. These hashes can then be used to prove the integrity of a device to a remote server through TIMA Attestation.

When unauthorized boot components are loaded, Trusted Boot will react in one of two ways. Low level components that are tightly tied to the device hardware, such as the bootloaders, should never be replaced; therefore, any attempt to replace these components produces a screen telling the user to take the device to a service center.

However, if the kernel has been modified, Trusted Boot instead sets the KNOX warranty violation fuse. This one-time programmable memory fuse indicates that the device has been tampered with and cannot use certain KNOX features thereafter. Additionally, even if the boot code is restored to its original factory state, this evidence of tampering remains and is reflected in the attestation results. Note that some device models will opt to never set the warranty violation fuse, instead always requesting that the user take the device in for service.

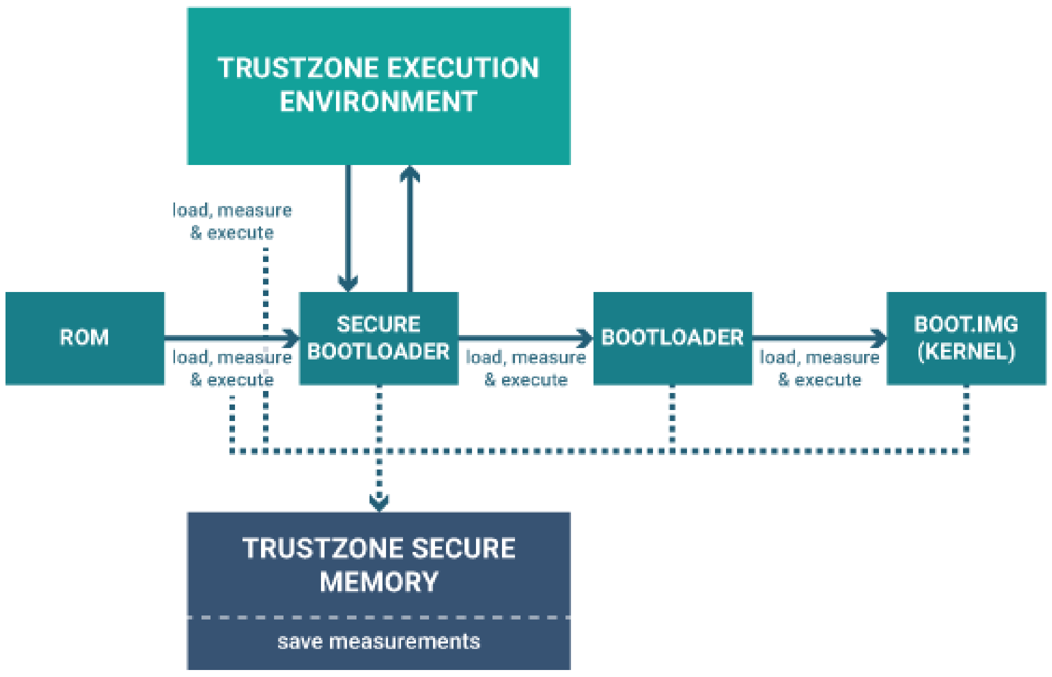

Figure 4 - The Trusted Boot Process

As each boot loader measures and executes the next, the measurements are stored in TrustZone secure memory for later inspection, i.e., through attestations.

Rollback Prevention (RP)

Rollback Prevention blocks the device from loading or flashing an approved but old version of boot components. Old versions of software may contain known vulnerabilities that attackers can exploit. Rollback prevention checks the version of the bootloader and kernel during both boot and updates, and blocks these processes from continuing if versions are unacceptably old. The lowest acceptable version of the bootloader is stored in secure hardware fuses when the device is flashed, and the lowest acceptable version of the kernel is stored in the bootloader itself. Whenever a vendor-applied update occurs, the lowest acceptable version can be incremented in the fuses. Because this value is kept in fuses, it cannot be decremented even through physical tampering.

Maintaining trust

Periodic Kernel Measurement (PKM)

TIMA PKM performs continuous periodic monitoring of the kernel to detect if legitimate kernel code and data have been modified by malicious software. In addition, TIMA also monitors key SE for Android data structures in OS kernel memory to detect malicious attacks that corrupt them and potentially disable SE for Android.

Real-time Kernel Protection

The security of the kernel is essential to the security of the whole system. An attack that compromises the kernel has the ability to arbitrarily access system sensitive data, hide malicious activities, escalate the privilege of malicious user processes, change the system behavior or simply take control of the system. As mentioned previously, Trusted Boot measurements can be used to determine what kernel was loaded and run when the device was started. However, this protection does not guarantee the integrity of the kernel after the system runs and starts to interact with potential attackers. Clever attackers can sometimes exploit an already booted and running kernel. In such cases, it is important to continuously monitor the kernel during the system *runtime* in order to detect and prevent modifications to the kernel code or critical data structures.

Intuitively, the kernel protection mechanism cannot itself exist completely in the kernel, or it could be circumvented by an attacker. Therefore, Samsung KNOX introduces Real-time Kernel Protection (RKP), a unique solution that provides the required protection using a security monitor located within an isolated execution environment. Depending on the device model, this isolated execution environment is either the Secure World of ARM TrustZone or a thin hypervisor that is protected by the hardware virtualization extensions. RKP's Trusted Computing Base (TCB) is part of this isolated environment and thus is secure from attacks that may potentially compromise the kernel.

Running in an isolated execution environment may cripple the ability of the security protection mechanisms to closely monitor events that happen inside the target kernel. To solve this problem, RKP uses special techniques to take full control over the normal world memory management and intercept critical events and inspect their impact on security before allowing them to get executed. Hence, RKP complements TIMA-PKM's periodic kernel integrity checking, which has limited effectiveness against attacks that can take place and properly hide their traces between periodic checks.

TrustZone-RKP achieves three important security features:

- First, it completely prevents running unauthorized privileged code (i.e., code that has the kernel privilege) on the system, which is accomplished by preventing modification of the kernel code, injection of unauthorized code into the kernel, or execution of the user space code in the privileged mode.
- Second, it prevents kernel data from being directly accessed by user processes. This includes preventing double mapping of physical memory that contains critical kernel data into user space virtual memory. This is an important step to prevent kernel exploits that map critical data regions into malicious processes user space so it can modified by an attacker.

- Third, RKP monitors some critical kernel data structures to verify that they are not exploited by attacks. In particular, RKP protects the data that defines the credentials assigned to running user processes to prevent attackers from escalating this credential by modifying this data.

NOTE: *The first feature is present on all models, while the second and third features are available on select models. Additional protection features are under development.*

Architecture overview

Figure 5 shows the architecture of RKP, which is hosted in an isolated execution environment that is protected even if Android's Linux kernel is compromised. The kernel is forced it to request RKP to perform two operations on its behalf: (1) emulating control instructions that change the system state and (2) updating the normal world memory translation tables. This monitoring is enforced by depriving the kernel of its ability to control these critical functions.

System control instructions allow the normal world to control security critical system state, such as defining the location of memory translation tables and exception handlers. These instructions can be only executed by privileged code, such as the kernel code. RKP instruments the kernel so that certain system control instructions are removed from its executable memory, which is the only memory that can execute privileged instructions in the normal world. Therefore, the only way to execute these instructions is through emulating them from the Secure World. We call this operation Control Instruction Emulation. On models that use the virtualization extensions, intercepting system control instructions can also be done using the hardware virtualization extensions.

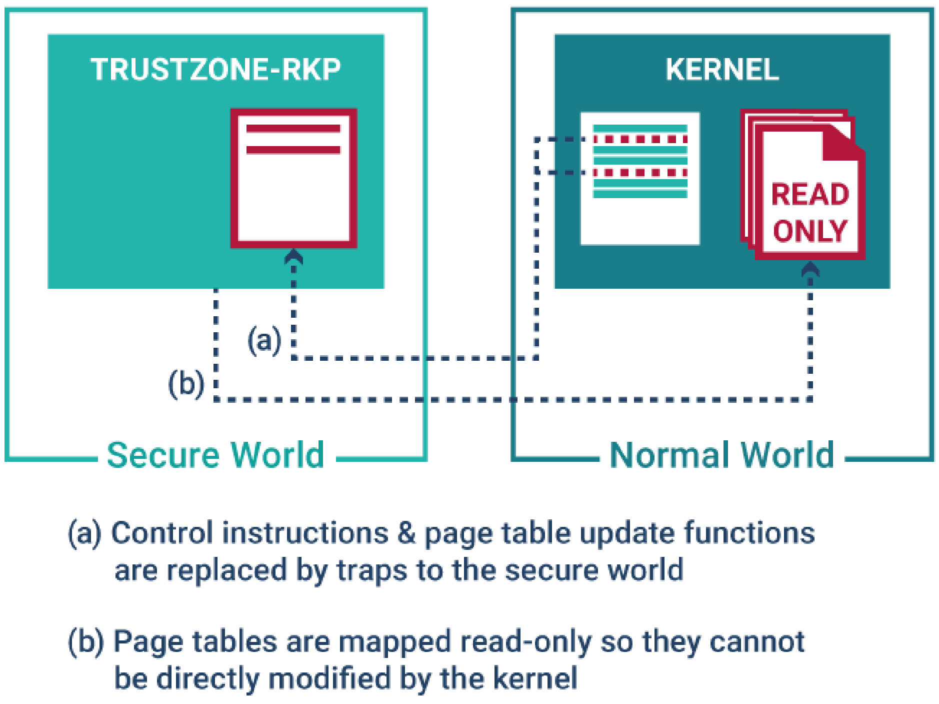
TRUSTZONE-RKP
KERNEL
READ ONLY
(a)
(b)
Secure World
Normal World
(a) Control instructions & page table update functions are replaced by traps to the secure world
(b) Page tables are mapped read-only so they cannot be directly modified by the kernel

Figure 5 - RKP Architecture (On select models, RKP runs in the virtualization protected environment rather than TrustZone Secure World).

Memory translation tables (also called page tables), define the virtual to physical address mapping and the access permissions of virtual memory. If the kernel attempts to change the current memory layout of the system through modifying translation tables, then RKP inspects these changes to confirm that they do not impact the system security. RKP ensures that translation tables cannot be modified by the normal world through making them read-only to the normal world kernel. Hence, the only way for the kernel to update the translation tables is to request these updates from RKP. As a result, RKP guarantees that this interception is non-bypassable.

Kernel code protection

Kernel code protection is the main security feature that RKP provides. The main guarantee is that an attacker that can get past the Linux kernel defenses would not be allowed to modify the kernel executable code, which greatly reduces the impact of kernel attacks on the whole system. To achieve this objective, RKP examines memory translation table modifications to enforce a set of rules that guarantee that the kernel is not writable by any code in the normal world. These rules also guarantee that the RKP monitoring cannot be bypassed even if an attacker finds a way to break the normal world kernel protections. Thus, the kernel is not able to modify its own code, even if it is compromised.

The rules are:

1. Kernel code pages are never mapped writable
2. Kernel data pages are never mapped executable
3. All memory translation tables are mapped read-only to the normal world
4. No double mappings of kernel code or any memory translation tables is allowed (Double mapping happens when the same physical memory is mapped to multiple virtual memory addresses, which might allow two different parts of the system to access the same memory with different permissions)
5. All mapped memory regions should have the Privileged eXecute Never (PXN) permission, with the exception of the OS kernel

The first two rules guarantee that the initial image of the kernel, as measured by Trusted Boot, cannot be directly modified by any potential attacker, unless it changes the memory mapping of the system by modifying the memory translation tables. This feature is still true even if the attacker takes control of the kernel itself. The rest of the rules guarantee that the memory translation tables themselves cannot be modified by the kernel, unless it sends a request to RKP. When such a request is sent, RKP verifies that the memory translation table modification does not violate the above rules. Combining these two sets of rules together, the kernel is not modified without RKP's knowledge.

These protections still require a basic assumption that the system memory management state has not been modified. Modifying the memory management system state (e.g., through changing the effective memory translation tables' base address or disabling the virtual memory protection completely) may allow an attacker to bypass the RKP monitoring. Thus, RKP uses the **Control Instruction Emulation** feature explained above to inspect these events to guarantee that they do not tamper with its monitoring.

In models that use the virtualization extensions, system control instructions are forced to trap into RKP through hardware controls. In models that use the TrustZone-based solution, this feature is complicated by the fact that TrustZone is not capable of trapping changes to the normal world state. Hence, RKP instruments the kernel to remove all instances of these system control instructions. Since these instructions can only run from privileged code, and RKP grants that privilege exclusively to the measured and protected kernel code, then it is absolutely impossible for the normal world to run these instructions without trapping to RKP. In turn, RKP validates the values to be written to the system control instructions to guarantee that they do not invalidate its kernel code protection assumptions.

Preventing double mapping of kernel data

Kernel data structures are critical to the security of the system. Maliciously modifying kernel data can lead to wide range of damage from privilege escalation to user process hiding. Since RKP completely protects the kernel code base and prevents return-to-user attacks using the PXN protection of user pages, there are only two possible methods to exploit kernel data. The first is through double mapping the memory hosting kernel data into the address space of the malicious process. The second is to alter the kernel control flow so that it maliciously modifies its own data (such as using pointer manipulation or pointer overflow).

The first class of attacks, double mapping of kernel data to malicious user processes, is a real threat to the kernel. For instance, a real-world Android exploit used an integer overflow to trick the kernel into mapping a huge range of the physical memory into the address space of the attacking process.

To prevent malicious double mapping of the kernel data, RKP ensures that physical memory pages hosting this data are not mapped to user space processes. They can only be mapped as privileged pages that cannot be accessed by the user space. RKP enforces this rule using its control of the normal world memory translation. RKP rejects any page table modification that maps kernel data to user space. To handle a related problem, RKP makes sure that no executable kernel pages are ever double-mapped to be writeable, and vice versa.

RKP relies on the target kernel to inform it about the location of its critical data. RKP embeds hooks inside the kernel code so it is informed whenever a new memory area is going to be allocated to the kernel. It then prevents this memory from being double mapped to writable memory anywhere else in the system.

This protection is effective against attacks that use double mapping to exploit kernel data. Although RKP relies on the kernel to inform it about the allocated data memory areas, this dependency does not weaken the protection. The kernel is assumed to be secure when it sends the information to RKP because this happens before the data pages are allocated. Afterwards, RKP prevents the data from being modified, except by the kernel itself.

Protecting the kernel data that defines user process credentials

After preventing kernel code modifications and double mapping of kernel data, the last class of attacks that threatens the kernel security is to alter the kernel control flow so that it maliciously modifies its own data. These attacks may include pointer manipulation, pointer overflow or return-oriented attacks.

Although RKP cannot fully protect against this class of attacks, it implements a novel technique to mitigate their effect through protecting selective kernel data structures that are critical to the system security. The data structure of choice is the process credentials data structure, which define the privilege level of the user processes running inside the device. User processes represent different running applications, such as user apps. In Linux, there is an instance of the credentials structure that is associated with each running process. This is frequently the target of rooting attacks, as by modifying this, a normal process can elevate its privilege.

RKP implements a three-step solution to protect the credential structure from malicious modifications. First, RKP makes all instances of the credential data structure read-only through controlling the memory translation tables. Second, it instruments the kernel so that all writes to the credential structures would be routed through RKP. This is guaranteed by the fact that the kernel now would not be able to write to this data from within the normal world. Before writing to the credential data, RKP examines the values to be written to make sure that they do not maliciously escalate the privileges of their corresponding user process. Determining if a user process is legitimately entitled to an escalated privilege, such as the administrative privilege, is done through combining multiple techniques. For instance, RKP prevents processes that start with regular user privilege from escalating their privilege after they start. In another example, processes that are started by applications that interface with potential attackers, such as zygote and adb shell, are not allowed to have an escalated privilege. Finally, RKP adds a check to the kernel security hooks to verify that a credential structure actually belongs to the read-only memory protected by RKP before it is effectively used to determine the privilege of the user process. Hence, it is guaranteed that a potential attacker cannot forge a malicious instance of the credential structures that is not monitored and verified by RKP.

For detailed information on RKP and the TrustZone-based implementation of RKP, see the following link on the ACM Digital Library website: http://dl.acm.org/citation.cfm?id=2660267.2660350&coll=DL&dl=GUIDE&CFID=629439201&CFTOKEN=91386218

DM-Verity

Attackers may not only be interested in attempting to modify bootloader or kernel images. There are many other software binaries and configuration files in storage which provide malware the property of persistence. Persistent malware is able to restart itself each time the system is rebooted. It does this by modifying programs or configurations on the system partition, which contains the system binaries, Android framework, and configuration files, that are started during boot. Once inserted into the boot path, the malware can survive system reboots. Additional problems can arise from tampering with system data and configurations, such as the granting of excessive privileges to vulnerable applications.

To prevent unauthorized modifications to the system partition, KNOX integrates a customized implementation of DM-Verity, a Linux/Android kernel module that performs integrity checks on all data blocks contained in a block device (such as a partition). In stock Android, DM-Verity uses a hash tree to perform integrity checks of individual data blocks. The root of the hash tree is signed by an RSA key. Whenever a data block is read into memory, DM-Verity computes the hash of the block, and then uses it, along with the other hashes on the path to the root to compute the root hash. If this computed root hash matches the signed version, the block is considered good. Otherwise, unauthorized modification of the block is detected, and the access to the data block is restricted.

KNOX's implementation of DM-Verity differs from stock Android in supporting file-based firmware over-the-air (FOTA) software updates. This approach is easier to support with the existing infrastructure than the stock block-based approach.

Proving trust

TIMA Attestation

TIMA Attestation allows a device to attest facts about its state to a remote server, such as an MDM server. The attestation message contains state measurements that can be evaluated by a server, which can then decide whether to trust the device or not. This message contains:

- Measurements collected by Trusted Boot to prove that only approved system software was loaded during boot
- Security violation logs from PKM and RKP since the last reboot
- Status of the KNOX warranty violation fuse
- Device-identifying information such as the IMEI and Wi-Fi MAC address
- A locally-computed verdict whether the device believes it is in a trustworthy state

The full attestation message is computed in the ARM TrustZone Secure World, and thus is accurate even if the entire normal world OS is compromised. Part of this attestation message is the verdict. Only when a) the measurements collected by Trusted Boot match known good values, and b) the warranty violation fuse is intact, the verdict is set to Yes to indicate attestation passed. The known good measurement values are kept in a file called tima_measurement_info, which is kept in TrustZone secure storage. This file is generated at build time. To simplify the logic of remote servers, they can directly use the verdict instead of verifying all measurements themselves.

To ensure unforgeability, the attestation message is signed using the TIMA Attestation Key, which is traceable to Samsung's root key. Each Samsung device supporting TIMA attestation has a unique RSA key pair, the device root key (DRK). The DRK is generated during manufacture, is traceable to Samsung's root key using X.509 certificates, and is stored in TrustZone. The remote server can verify the integrity of this message using Samsung's root key. To ensure that the attacker of a compromised device cannot replay old valid attestation messages, the signature includes a server-generated cryptographic nonce, which is a random number used only once.

To illustrate the use of this capability, consider the MDM server example again. Depending on the attestation verdict and the data, any further action is determined by the enterprise's MDM security policy. The security policy might choose to detach from the device, erase the contents of the secure Workspace, ask for the location of the device, or any of many other possible security recovery procedures.

Part 2. Making the trusted environment enterprise ready

Section 1 described how KNOX built a trusted environment where the integrity of its components is tied back to hardware. The subsequent sections describe technologies built on top of this trusted environment to enable KNOX for enterprise use.

SE for Android

Samsung KNOX adopts Security Enhancement for Android (SE for Android), which adds Mandatory Access Control (MAC) to Android. Many people are aware of Discretionary Access Control (DAC) mechanisms, such as Android permissions or Linux owner/group/world permissions. DAC mechanisms have limited security benefits since the user or process generating data has discretion to change the access rules for that data. A user can make bad decisions with data, which may then be leaked publicly. MAC is designed to let security experts enforce rules that can't be maliciously or ignorantly overridden by device users or software developers. Since these rules are mandatory, and cannot be altered by users or developers, they provide a way to prevent malicious code or untrusted users from accessing sensitive data or programs. MAC can be used to lock down data that a user wants to keep secret, and prevents developers from maliciously or accidentally compromising the system components that protect our devices.

SE for Android provides two layers of MAC protection:

1. **Kernel-level protection**: Android inherits the SELinux MAC abilities directly from Linux. SELinux provides MAC for kernel system calls. SELinux policy can enforce which objects these system calls can target. For example, you can specify in SELinux policy that only system-signed processes can read files in the directory/data/security. This level of control is possible because access check hooks are inserted inside the kernel. These hooks query the security policy before each system call to determine if it represents an allowed action. SELinux policies can prevent processes from reading or tampering with data, bypassing security mechanisms, or otherwise interfering with other processes. They also limit the damage from malicious or flawed programs.

2. **Android middleware protection**: There are many parts of the Android system that don't leverage system calls to get things done. An example would be the Android Intents used to start apps. This layer of software above the kernel, but below user space applications, is called the Android middleware. Additional hooks have been added to key decision points to extend MAC control to the middleware. This is known as Middleware MAC (MMAC). MMAC can enforce security policies among inter-component communication for Android Apps.

The main security objectives of SE for Android include strong data and application isolation, confining the permissions of system processes running as root, and protecting applications.

Scope of Access Control

Samsung's custom version of SE for Android provides the following unique features:

- MAC on APIs (control who can call your APIs)
- KNOX Workspace isolation of personal & business data
- On-the-fly Workspace creation for customizing your security
- Quick-response policy updates (no carrier-approved firmware update required to plug many vulnerabilities)
- Strong application isolation beyond Android's standard access control
- Extensible MAC for new KNOX features

Samsung also built an innovative global policy validation system that can detect when prohibited actions are attempted. This gives us unique visibility into how our devices are used and can alert us to new threats. This system can be used to refine our policy and very accurately grant only the permissions needed.

SE for Android Policy

SE for Android includes a set of security policy configuration files designed to meet common, general-purpose security goals. Out of the box, Samsung KNOX provides a policy that is designed to strengthen the core Android platform and meet enterprise needs. Samsung KNOX also offers the SE for Android Manager Service (SEAMS), which provides management APIs that allow enterprise IT admins to manage SE for Android. Management tasks include gathering access logs, resetting file security labels, mapping applications to different security domains, getting type context information, and getting status information about packages and Workspaces.

For enterprise apps that run in a Workspace, Samsung KNOX provides policies to enforce the isolation of application Workspaces. For example, Samsung KNOX has created new security domains and can also now enforce Multiple Category Security (MCS) isolation. Categories are used to isolate applications and data into security groupings, independent of what security domain they are assigned. Categories can then be used to ensure that personal applications and business applications with the same security domain have their access rights limited to their own areas. New Workspaces can be created on the fly without having to edit the security policy by simply applying a new security category to a group of apps.

KNOX Workspace: divide and conquer

According to a Workshare survey conducted this year, 800 employees from around the world agreed by 65% that they are responsible for protecting company data. However, 80% of those surveyed are using unsecured file sharing methods, which puts corporate data at risk.[6]

The Samsung KNOX Workspace provides a separate Android environment within the mobile device, complete with its own home screen, launcher, applications, and widgets. This separation gives the enterprise control of securing sensitive data and intellectual property, while providing employee privacy in the personal environment. Separating the work and personal environments is the best way to prevent malware from being downloaded into the Workspace.

Applications and data inside Workspace are isolated from applications outside Workspace, that is, applications outside Workspace cannot use Android inter-process communication or data-sharing methods with applications inside Workspace. For example, photos taken with the camera inside Workspace are not viewable in the Gallery outside Workspace. The same restriction applies to copying and pasting. When allowed by IT policy, some application data such as contacts and calendar data can be shared across the Workspace boundary. The end user can choose whether they prefer to share contacts and calendar notes between Workspace and personal space; however, IT policy ultimately controls this option.

The enterprise can manage Workspace like any other IT asset using an MDM solution; this container management process is called Mobile Container Management (MCM). Samsung KNOX supports many of the leading MDM solutions on the market. MCM is affected by setting policies in the same fashion as traditional MDM policies. Samsung KNOX Workspace includes a rich set of policies for authentication, data security, VPN, e-mail, application blacklisting and whitelisting, among others.

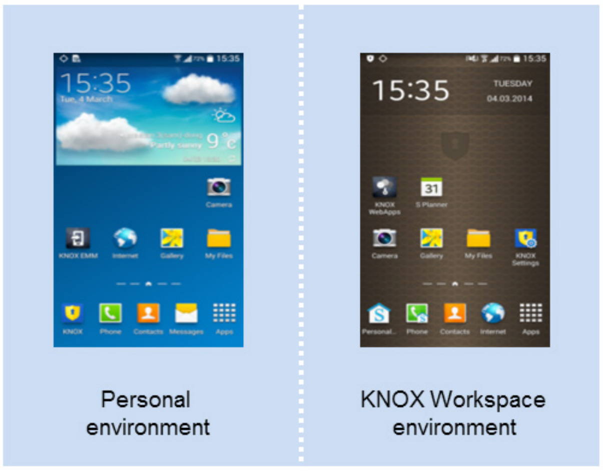

Figure 6 - User's personal environment running next to the Workspace environment

The KNOX 2.X platform features the elimination of application wrapping, which was used by KNOX 1.0 and many other competing solutions. This is achieved by leveraging technology introduced by Google in Android 4.2 to support multiple users on devices. It reduces the barrier to entry for independent software developers wishing to develop and deploy applications for KNOX Workspace.

KNOX Workspace can also be configured for container-only mode. In this mode, the entire device experience is restricted to the Workspace. This mode is suitable for industries such as health care, finance, and others who provide devices for employees that seek to restrict access to business applications.

Workspace also supports a two-factor authentication process. The user can configure the Workspace to accept a fingerprint as the primary authentication factor for the container with a PIN, password or pattern as a second factor.

The KNOX platform supports two containers, thus meeting the needs of professionals that use their own devices for corporate use (BYOD) and have multiple employers, such as doctors or consultants.

New features include the ability to enable Bluetooth and Near Field Communication (NFC) inside Workspace. External SD cards can also be enabled with security restrictions.

Sensitive Data Protection

KNOX can enforce two classes of protection for data generated from within the KNOX Workspace: protected data and sensitive data. All data generated from within the KNOX Workspace is considered to be protected. Protected data residing in storage is always encrypted, and is thus protected against offline attacks, e.g., forensic analysis on a flash memory image extracted from a stolen device. Furthermore, access controls are used to prevent applications outside the KNOX workspace from attempting to access protected data. The decryption key for protected data is stored encrypted by the device-unique hardware key (DUHK). Therefore, the key is only recoverable on the same device.

Sensitive data, on the other hand, provides an even stronger security guarantee. Like protected data, sensitive data is always encrypted when on disk. Additionally, the data remains encrypted as long as the Workspace is locked. The key used to encrypt sensitive data on disk is recoverable only if the user enters the Workspace password, PIN, or pattern. Thus, if a device is stolen, the key cannot be extracted from anywhere on the device. As with protected data, the stored key material is encrypted by the DUHK, thus binding it to the device.

Enforcement of this guarantee for sensitive data is performed by KNOX Sensitive Data Protection (SDP). SDP creates a Container Master Key (CMK) that can only be decrypted with user input. If desired, the MDM (see the section Mobile Device Management) can also be used to unlock the CMK, thus preventing total data loss in the event of a forgotten Workspace password. Once the Workspace is locked, SDP clears all keys in memory after a configurable timeout (five seconds by default). In addition, SDP also flushes sensitive file data from the OS kernel's disk caches if the file is not in use by a Workspace application.

Any sensitive data received when the Workspace is locked will still be protected by SDP. This works by using a public key algorithm in which the private part of the key is maintained in an encrypted partition, and the public part is used to encrypt the new sensitive data. Once the Workspace is unlocked, the data is decrypted with the private key, and re-encrypted using the usual symmetric key, which is guarded by the CMK. Currently, email subjects, bodies and attachments are marked sensitive. Additionally, the SDP Chamber provides a directory, in which all files are automatically marked as sensitive, and protected by SDP.

On-Device Encryption

The KNOX platform further strengthens the full-device encryption capability offered by the Android platform. In addition to Android's kernel-level full device encryption, KNOX ties the encryption key to a secret maintained in trusted hardware. This feature is available only if the enterprise IT administrator activates encryption via the MDM. TrustZone-based on-device encryption (ODE) also enables enterprises to ensure that all device data is protected in the unlikely event that the operating system is compromised. While this feature is low overhead, providing system-wide encryption means less flexibility in supporting separate security levels for user and enterprise data, hence the inclusion of the finer-grained protected and sensitive data classes.

Trusted Boot Based KeyStore (TIMA KeyStore)

The TIMA KeyStore provides applications with services for generating and maintaining cryptographic keys. The TIMA KeyStore is only enabled if the Trusted Boot measurements match the known good values in the file tima_measurement_info, and if the KNOX warranty fuse is not set. Thus, cryptographic operations with keys in the KeyStore can only occur if the system was booted into an approved state. Keys stored in the TIMA KeyStore are further encrypted with the device-unique hardware key (DUHK), and can only be decrypted from within TrustZone Secure World on the same device. All cryptographic operations on the keys are performed within TrustZone Secure World.

The TIMA KeyStore has the same API as the familiar Android KeyStore APIs. Therefore, the only modification necessary is to specify that the TIMA KeyStore be used to provide the service.

Trusted Boot Based Client Certificate Management (TIMA CCM)

The TIMA CCM enables storage and retrieval of digital certificates, as well as encryption, decryption, signing, and verification in a manner similar to the functions of a SmartCard. The certificates and associated keys are encrypted with a device-unique hardware key that can only be decrypted from code running within TrustZone.

TrustZone-based CCM also provides the ability to generate a Certificate Signing Request (CSR) and the associated public/private key pairs in order to obtain a digital certificate. A default certificate is provided for applications that do not require their own certificate.

Programming interfaces for certificate storage and management are provided in the KNOX Premium SDK. Application developers are provided with industry standard PKCS #11 APIs for certificate management, and therefore interact with the CCM as if it were a virtual SmartCard. Like the TIMA KeyStore, TIMA CCM operations are permitted only if the device was booted into an approved state.

Trusted UI

Another service required by many enterprise applications is some form of authentication. For enterprises wishing to use PIN-based authentication, KNOX provides the Trusted UI for secure credential entry. The *Trusted UI* uses ARM TrustZone to create a dedicated path through hardware from the screen and keyboard to the Secure World. Any credentials entered while this path exists will be completely inaccessible to normal world programs and untrusted peripherals. Once the credentials are held in the Secure World, they are passed back to the enterprise application that initiated the authentication.

Virtual Private Network

KNOX provides a rich set of VPN features to address a wide-range of enterprise mobile device deployment scenarios. At the core of KNOX VPN framework is the Generic VPN Service that enables VPN vendors to provide a wide range of features and configurability.

VPN features of KNOX include:

- Administrator-configured System VPN
- Administrator-configured Per-App VPN
- Administrator-configured Workspace VPN
- Multiple concurrent VPN connections
- IPsec and SSL VPN support
- Administrator-configured FIPS and non-FIPS VPN mode
- Common Access Card (CAC)-based authentication
- Always on VPN connections with auto-reconnect
- VPN tunnel chaining

VPN connections in KNOX are managed by MDMs through GenericVPNPolicy class. This class provides APIs for an MDM Admin application to configure VPN settings, certificates, and applications.

KNOX VPN partners implement the KNOX Vendor SDK APIs to enable rich management capability for MDMs. A number of leading VPN vendors have released VPN clients for KNOX.

In addition to traditional VPN clients, other network packet processing applications can be enabled on a KNOX platform. Network traffic optimization, split billing and network access control are a few examples of applications that integrate with the KNOX VPN framework. The KNOX VPN framework ensures proper chaining of the traffic when multiple applications are configured to process network packets.

System, Per-App, and Workspace VPN

MDM vendors can configure the following kinds of VPN profiles:

- A System VPN profile affects all traffic from the device. There can only be one System VPN profile in effect at a time. However, per-app VPN and Workspace VPN profiles can coexist with a System VPN profile.

- Workspace VPN profiles affect network traffic for a given KNOX Workspace. Applications outside the Workspace are not included in the VPN tunnel.

- Per-App VPN profiles affect one or more named applications. All network traffic for the configured applications is tunneled through the VPN connection.

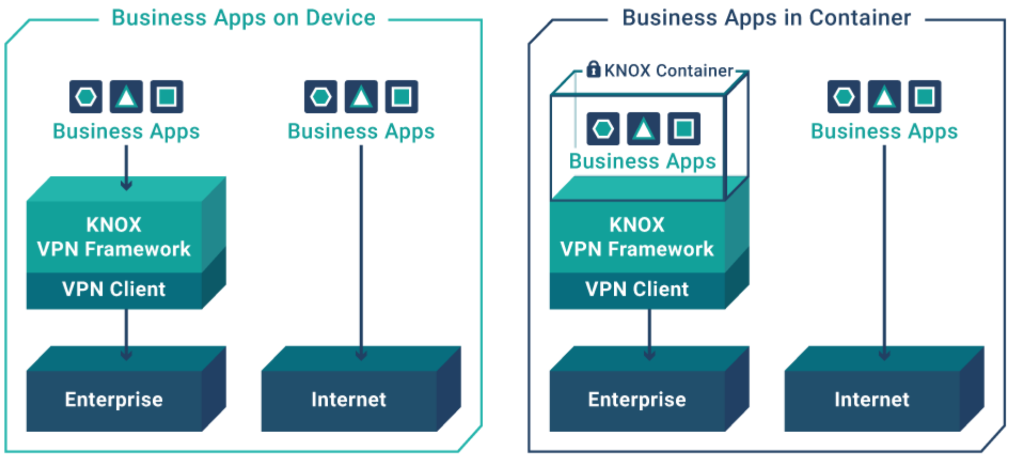

Figure 7 - Per-app VPN

For any of the VPN profiles described, all traffic for a given application is tunneled through the VPN, if it is included in the profile, irrespective of the destination address of the network packet. For example, if the browser is included in one of the VPN profiles, all browser traffic goes through VPN, whether or not the destination URL is inside the corporate network or not.

When a System VPN profile is activated along with a per-app VPN or Workspace VPN, the MDM administrator can determine the behavior as below:

- Per-App VPN or Workspace VPN connections are tunneled through the System VPN (for example chaining), or,
- Applications that are not already included in other active VPN profiles are included in the System VPN tunnel.

VPN Auto Reconnect and Always On

The Auto Reconnect feature allows failed VPN connections to be automatically restored depending on the failure condition.
Some of the key features of Auto Reconnect design are:

- When a VPN connection fails because of network/timeout reasons, connections are automatically retried.
- The Auto Reconnect feature can be enabled/disabled by the MDM administrator.
- A user can be given the option to manually retry the failed connection.

It's also possible to configure the VPN to be Always On. In this case, the VPN tunnel is automatically established after the phone boots, and remains on. Configured applications are tunneled through the VPN (whether System, Per-App, or Workspace).

SmartCard framework

The United States Department of Defense (US DoD) has mandated the use of Public Key Infrastructure (PKI) certificates for employees to digitally sign documents, encrypt and decrypt e-mail messages, and establish secure online network connections. These certificates are typically stored on a SmartCard called the Common Access Card (CAC).

The Samsung KNOX platform provides applications access to the hardware certificates on the CAC via standards-based Public Key Cryptography Standards (PKCS) APIs. This access process enables the use of the CAC card by the browser, e-mail application, and VPN client, as well as other custom government applications.

Other enterprises have growing interest to use SmartCards for the same purpose, especially those that require high levels of security and information protection.

The KNOX platform provides improved SmartCard compatibility via a software framework that allows third-party SmartCard and reader providers to install their solutions into the framework.

Single Sign-On

Single Sign-On (SSO) is a feature that provides common access control to several related, but independent, software systems. The user logs in once and has access to all systems without being prompted to log in again. For example, SSO allows access to the container (and participating apps that require credentials within the container) with one password.

SSO shares centralized authentication servers that all other applications and systems use for authentication purposes. It combines this with techniques to ensure that users do not have to actively enter their credentials more than once.

Advantages of using SSO include:

- Reduces the number of user names and password combinations a user must remember
- Reduces time spent re-entering passwords for the same user
- Reduces IT costs with fewer help desk calls about passwords
- Increases security because tokens and certificates are transmitted over the internet for authentication as opposed to plain text passwords

Active Directory Integration

KNOX now provides an option for the IT admin to choose an Active Directory password as the unlock method for KNOX containers. This has two important benefits. First, it allows IT Admins to use a one-password management policy for desktop and mobile devices. Second, the end user only needs to remember one password to access all services offered by the employer, thereby reducing employee password fatigue and improving productivity.

At the heart of this feature is the proven industry-standard Kerberos protocol. Active Directory is the most widely-deployed enterprise grade directory service that has built-in support for Kerberos. KNOX provides a set of Workspace creation parameters to configure Workspace to use the active directory password as the unlock method. Additionally, IT Admins can also configure Single Sign-On for services inside Workspace, along with the unlock method.

Mobile Device Management

KNOX provides hundreds of Mobile Device Management (MDM) security policies for fine-grained control of devices. The solution includes:

- Mobile Device Management (MDM)
- Mobile Application Management (MAM)
- Identity and Access Management (IAM)

The broad categories of supported MDM APIs are shown in Table 2.

KNOX MDM policies are designed to lower cost and improve usability and manageability for small or medium enterprises. The full mobile and web application solution has cross-platform support for Samsung devices, other Android devices, and iOS™ devices to support BYOD or COPE.

Support for cross-platform devices creates a centralized location for enterprises to manage devices. Mobile Application Management focuses on data management, as well as who has access to applications.

Identity and Access Management adds another layer of security with automated user authentication and easy access for administrators to monitor all activity. IAM reduces password errors with convenient Single Sign-On (SSO) and gives IT Admins time to focus on policy enforcement.

Enterprises can use the cloud-based policy management, an on-premise Active Directory, or a hybrid combination to separate employees and external or partner users. The full mobile and web application solution has cross-platform support for Samsung devices, other Android devices, and iOS™ devices to support BYOD or COPE.

KNOX MDM API Categories

Enterprise IT Compatibility

- Account Management using blacklisting/whitelisting
- Active Directory integration
- LDAP Management

Security and Compliance

- Device Admin Management
- Firewall
- Password Management
- Device Security
- Remote Event Injection
- Audit Logging
- Usability
- Kiosk Mode
- Workspace Management
- Multi-user Mode

Device Control

- Date and Time
- Bluetooth
- Location Management
- Device Restrictions
- Wi-Fi Configurations
- APN Settings
- Device Inventory

Application Management

- Browser
- Email/Exchange Configuration
- Application Management

Telephony

- Telephony Management
- SIM Change Information
- Roaming Restrictions

Simplified enrollment

Enrolling an Android device into a company's MDM system typically begins with a user downloading the agent application from the Google Play store, then configuring it for authentication. Enterprises are facing increasing help desk calls as more and more users are activating mobile devices for work. When presented with prompts, privacy policies, and license agreements, users might experience difficulties during the process, resulting in a poor overall experience.

The KNOX platform provides a simplified enrollment solution that is streamlined and intuitive and eliminates many steps and human error.

The enrollment process provides the employee with an enrollment link sent by e-mail, text message, or through the company's internal or external website. Once the link is clicked, users are prompted to enter their corporate e-mail address. This action triggers the display of all required privacy policies and agreements. After accepting the terms, users enter a corporate account password for authentication from the enterprise. Any agent application required is automatically downloaded and installed.

MDM vendors can take advantage of this feature to simplify the onboarding process for enterprise users, significantly improve the user experience, and reduce support costs.

The KNOX platform offers significant enhancements to the management policies previously offered. In particular, bulk enrollment now allows IT Admins to enroll hundreds or thousands of employees at the same time. An IT Admin portal is used to create an MDM profile where the IMEIs of employees are added. Employees receive a notification to accept enrollment, or another method can seamlessly enroll users without user authentication.

In addition, the Samsung KNOX Mobile Enrollment allows IT Admins to enroll hundreds or thousands of employees at the same time. Samsung provides a web tool and an application to scan package bar codes. The device requires end user approval before enrolling to the MDM server.

KNOX Mobile Enrollment supports multiple MDM configurations per account. With complex device environments, and multiple MDM profiles or configurations, KNOX Mobile Enrollment gives IT Admins the ability to prepare hundreds of devices and get them connected to the right MDM with ease. End users only need to turn on the device and connect to the network. KNOX Mobile Enrollment takes care of activation without users needing to do a thing.

Enterprise split billing

The KNOX platform now supports Enterprise Split Billing, which now provides enterprises a mechanism to discriminate enterprise data usage from data usage. This enables enterprises to properly compensate their employees for the cost generated because of work, particularly in BYOD cases, or pay only for work-related data in COPE cases. The current KNOX Split Billing solution requires carrier cooperation in order to properly handle the billing information.

Endnotes

[1] Juniper Networks, "Juniper Networks Third Annual Mobile Threats Report, March 2012 through March 2013," p. 4-6. http://www.juniper.net/us/en/local/pdf/additional-resources/jnpr-2012-mobile-threats-report.pdf

[2] Aspect Security, Inc., "2013 Global Application Security Risk Report," p. 2. http://cdn2.hubspot.net/hub/315719/file-681702349-pdf/presentations/Aspect-2013-Global-AppSec-Risk-Report.pdf

[3] Nielsen, "The Digital Consumer," October 2013, p. 8. http://www.nielsen.com/content/dam/corporate/us/en/reports-downloads/2014%20Reports/the-digital-consumer-report-feb-2014.pdf

[4]Consumer Reports, "Smart phone thefts rose to 3.1 million last year, Consumer Reports finds," May 2014. http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm

[5] FCC, "Report of Technological Advisory Council (TAC) Subcommittee on Mobile Device Theft Prevention (MDTP)," December 2014, p. 22. http://transition.fcc.gov/bureaus/oet/tac/tacdocs/meeting12414/TAC-MDTP-Report-v1.0-FINAL-TAC-version.pdf

[6] Workshare, "Data Guardian: Detecting Business Risk 2014," p. 14-16. https://d3liiczouvobl1.cloudfront.net/uploads/refinery/resource/file_name/251/Workshare_-_Data_Guardian_-_Detecting_Business_Risk_2014.pdf

About Samsung Electronics Co., Ltd.

Samsung Electronics Co., Ltd. is a global leader in technology, opening new possibilities for people everywhere. Through relentless innovation and discovery, we are transforming the worlds of televisions, smartphones, personal computers, printers, cameras, home appliances, LTE systems, medical devices, semiconductors and LED solutions. We employ 236,000 people across 79 countries with annual sales exceeding KRW 201 trillion. To discover more, please visit www.samsung.com

For more information about Samsung KNOX, visit www.samsung.com/knox

Samsung Electronics Co., Ltd.
416, Maetan 3-dong, Yeongtong-gu
Suwon-si, Gyeonggi-do 443-772, Korea