Assertion Framework for BYOD

Chris Daly

General Dynamics C4 Systems

Chris.daly@gdc4s.com

Overview

- BYOD Problems, Requirements, and Scenarios
- What is an assertion?
- Why trust assertions for BYOD?
- Keys to understanding trust assertions
- BYOD trust assertion views covered
  - Life Cycle
  - Platform
  - Transaction
  - Inter-Context
- BYOD Scenarios – Revisited

Problems with BYOD

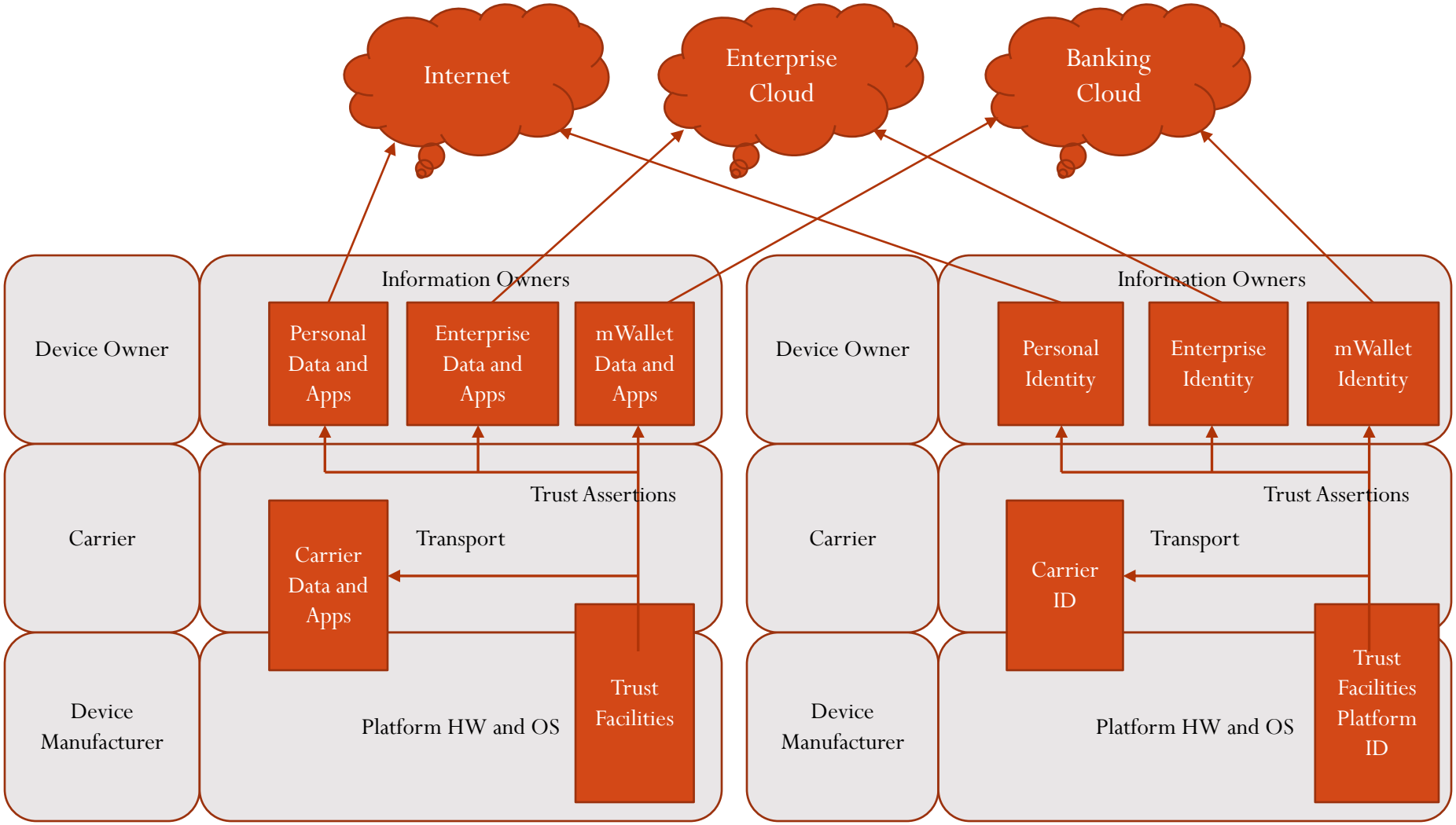
- Multiple personalities or entities on a single device can create security issues
- In general, many security problems are related to improper assumptions or poor management of trust relationships between interacting components or entities
  - Examples of problems: Social engineering/spear phishing, identity spoofing, domain hijacking, man-in-the-middle attacks, cross-site scripting
  - BYOD security should focus on the establishment and maintenance of trust assertions that manifest the operating state and reputation properties of entities involved in interactions
- BYOD interactions occur at many levels and cut across device and enterprise boundaries
  - "Trust" interactions involve different components of a device working together including information owners, processes, applications, data, and hardware elements
  - Transitive or federated trust situations create significant security exposures if trust is not established or verified at each node in the chain
  - Trust of an entity can be evaluated when you actually know its state, so the entities being trusted must be *measurable in a security and operations context*.

Requirements for BYOD

- From a proactive security perspective, *measurable* translates to:
  - Ensuring an optimally secure state for the device [or container] is identified for the usage of an asset across the life cycle of the asset
  - The current security state of the device [or container] can be determined and reported in a trustworthy way
  - The security status of a device {or container] is linked to the business processes, policies, and usage of that asset
- A "proactive" operations approach requires that critical assets are instrumented to produce security assertions of their security state, that these assertions are reported on a continuous basis, and monitored in relation to critical mission activities and threats that are germane
- Assertions should identify: changes that cause the device [or container] to be out of compliance (good); and changes or activities that indicate a breach or compromise (better)

BYOD Notional Scenarios

BYOD: Mobile devices that provide policy-managed, verifiable, and isolated contexts in which to process and store data and provide confidence to information owners in the proper protection of their data.



What is a Trust Assertion?

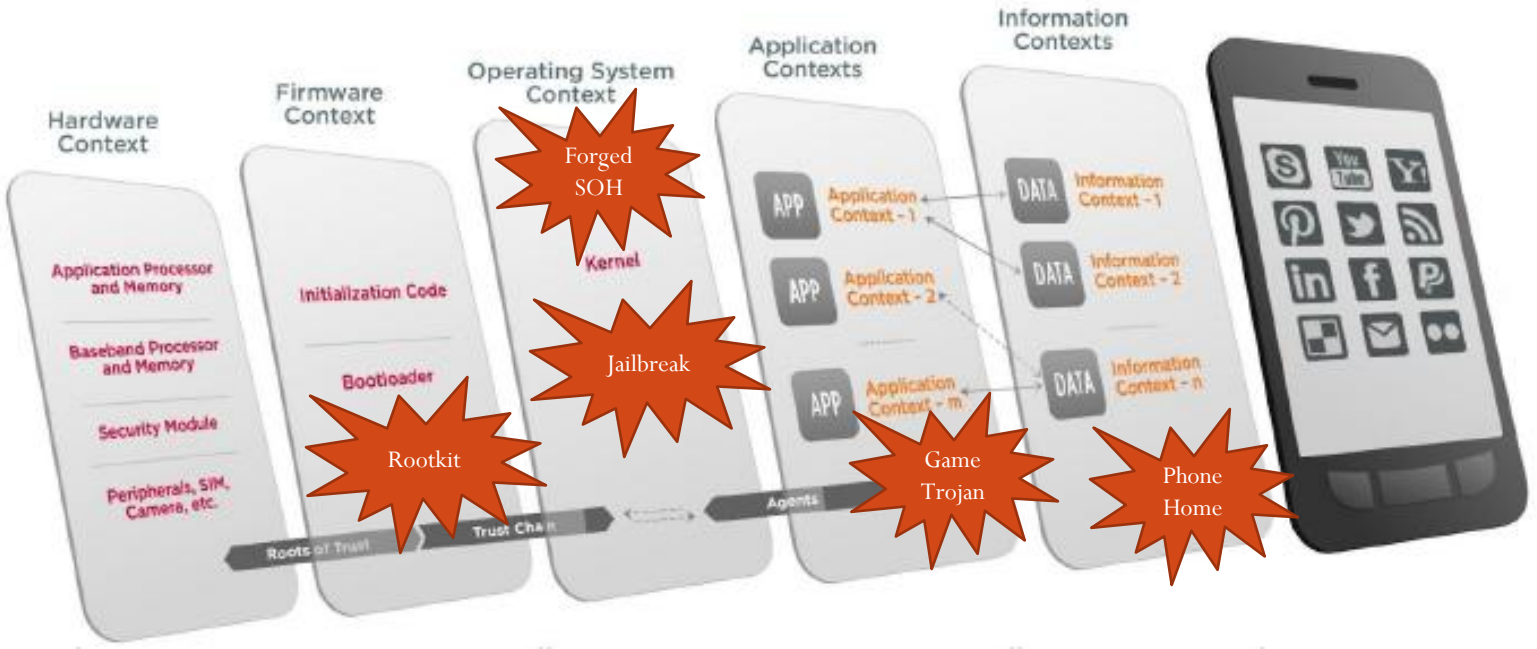
- An ***assertion*** is a set of one or more attributes that represents the state of an entity in a transaction, usually on-line, but could be between two different information domains on the same device
- Mobile devices may use assertions to represent the state of firmware as either verified or unverified, the state of an OS as either validated or not, the state of file encryption as either on or off, the state of the microphone as either on or off, etc.
- A Policy Enforcement Engine collects assertions from a mobile platform [or a virtual container on a platform] and determines whether or not the platform [or container] is in compliance with a particular policy.
- A "trust assertion" uses "trusted components" and reflects methods for positively establishing relevant properties of an entity, a secure path for communication, a secure space for processing, a secure method to prevent observation or interference, and a secure API for interactions

Why Trust Assertions for BYOD

Assertions provide a way for the Information Owner to detect if the Device Owner's activities have unknowingly altered the state of the device in such a way so as to put the Information Owner's data [context] at risk.

A Root of Trust (RoT) provides an anchor on which to base integrity assertions about the mobile device's configuration, health, or operating status that can be verified by the Information Owner through a remote or local verifier.

Standards (based on NIST SP 800-157) to format and perform the integrity measurement assertions enable scalable management of uniform policies possible even on devices not directly controlled by the Information Owner.



Examples of Trust Assertions

- Are the device and firmware you are using been modified? *I can attest that I booted securely and without unauthorized mods*
- Who are you and are you trusted? *I can assert unforgeable proof of my identity*
- Is the code or application you are using trusted? *I can assert proof that the software I am using is trusted based on this evaluation certificate*
- Has the information you are sending or receiving been tampered with and is it authoritative? *I assert that the information I sent you came from me and was not modified in transit.*
- Are you authorized to perform this process? *I can assert my authority based on this certificate of authority which is cryptographically based.*
- Do you need to perform this process? *I can assert a requirement to perform this process based on this chain of authority and certified mission priority (e.g., orders).*
- Is the infrastructure trusted that was used to support your request? *I can assert that the infrastructure is operating according to design based on this certification.*
- Is the organization you belong to trusted? *I can assert my organization is trusted to support this effort based on reputation and proven experience citations.*
- Are you in a "context" that is trustworthy commensurate with the sensitivity of the information and use of resources involved in the process (e.g., printer, browser)? *I can assert that the context in which I am operating is isolated from interfering or adversarial contexts, and that it is certified to handle and protect the sensitivity of data to be processed based on this certificate or cryptographically protected label.*

Conditions for BYOD Assertions

- Immutable Root of Trust: The Root of Trust (RoT) possesses a certain level of immutability so that the Information Owner can be confident that no matter what state the mobile device is in, the RoT has not been affected.
- Transitive Trust Chain: Assertions start from RoTs and work their way up the stack / various contexts based on a transitive trust chain.
- Continuous [Device/Context] Monitoring: There needs to be a way to constantly re-affirm that the device's (and specific IO context) state has not been compromised - the device must be able to make certain reliable non-repudiable assertions about itself to the information owner.
- Policy Flexibility: In some scenarios, information owners may want explicit assertions to be made about specific information contexts. Assertions may need to be compounded to build new assertions. Gradations of assertions may be needed.
- On-Board and Remote Assertion Reporting: Devices should possess the ability to make assertions both locally and remotely.
- Life Cycle Standards: The mobile device needs to be initially examined, configured, provisioned and affirmed as trusted by Device Manufacturer and Information Owner.

What is a Root of Trust

- A Root of Trust (RoT) is more than just a certificate; it is a computing element that executes a set of unconditionally trusted functions in support of device integrity, isolation, and protected storage.
- A RoT must always behave in an expected manner because RoT misbehavior cannot be detected.
- Hardware roots of trust are preferred over software roots of trust due to smaller attack surfaces and more reliable behavior.
- Roots of trust instantiated at the beginning of device power-up are immutable and are in a good position to serve as anchors for chains of trust for measurement and verification.
- RoTs must be exposed to the device and operating system in order to establish a chain of trust for user applications and to provide assertions.
- Mobile applications interacting with services offered by various information owners will frequently utilize the capabilities provided by the RoTs to locally store cryptographic keys, authentication credentials, and other sensitive data.
- Information owners will frequently rely on assertions based on RoTs.

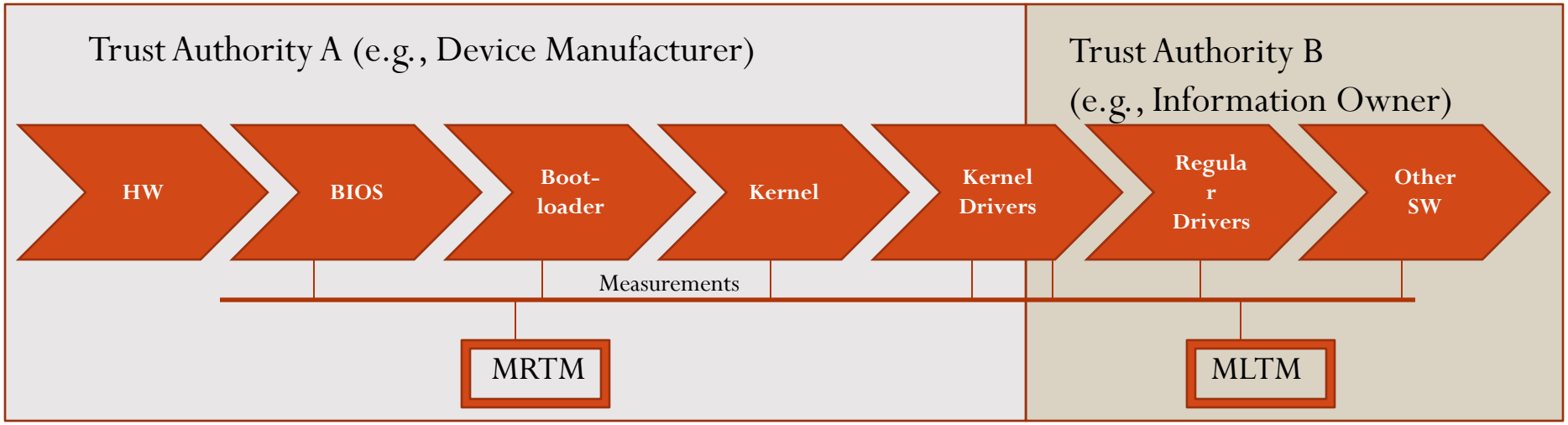
Mobility Roots of Trust

- **Root of Trust for Storage (RTS) -** provides protected repository and protected interface to store and manage keying material
- **Root of Trust for Verification (RTV) -** Works in conjunction with the RTM to verify and authenticate the integrity of all software and create assertions based on the result. It checks the software measurements against their reference values
- **Root of Trust for Integrity (RTInt) -** provides protected storage, integrity protection, and protected interface to store and manage assertions
- **Root of Trust for Identity (RTId)** - provides a protected interface to manage identities and sign assertions.
- **Root-of-Trust-for-Reporting (RTR) -** provides authenticity and non-repudiation services. The RTR is used to authenticate the identity of the sender of data and the source of the data attested.
- **Root-of-Trust-for-Measurement (RTM) - p**erforms the measurement functionality. The RTM should ensure that these integrity measurements are protected by an RTInt.
- **Root-of-Trust-for-Update (RTU) -** verifies the authenticity of signed updates, upon successful verification, initiate the update process. May be used to protect the other roots of trust.
- **Root-of-Trust-for-Enforcement (RTE) -** a trusted entity that builds any of the RoT components that are based on allocated resources. Examples of allocated resources include a trusted operating system or a function implemented in a software application. If all RoT components are provided as dedicated resources, the RTE is not required. Typically, the RTE is the trusted code that is stored in ROM and executed on platform reset to begin secure boot.

Root of Trust Principles

- Roots of trust can be implemented as a combination of hardware and software to provide the best balance of cost of hardware against the security provided. E.g. the mobile TPM provides RTId, RTR; the GP TEE provides RTS and RTInt; while the BIOS provides RTM, RTV, and RTU
- The security strength of trust roots is defined by properties like resilience to unauthorized modifications, cryptographic strength, method of platform bindings, and evaluation level in associated documents
- Modifications to a RoT must be either cryptographically authenticated from the RoT owner, or verified through the use of a secure local update mechanism (e.g. physical presence)
  - If the RoT is modifiable, it must have a uniquely and cryptographically identifiable owner
  - If the RoT can modify other RoTs, it must be uniquely and cryptographically identifiable
- A RoT cannot be verified before its execution, therefore it must be trusted to work correctly. Note: In the mobile world, the RTE is inherently trusted, and the other RoTs are verified before execution (and do not boot if they are not verifiable)

Transitive Trust



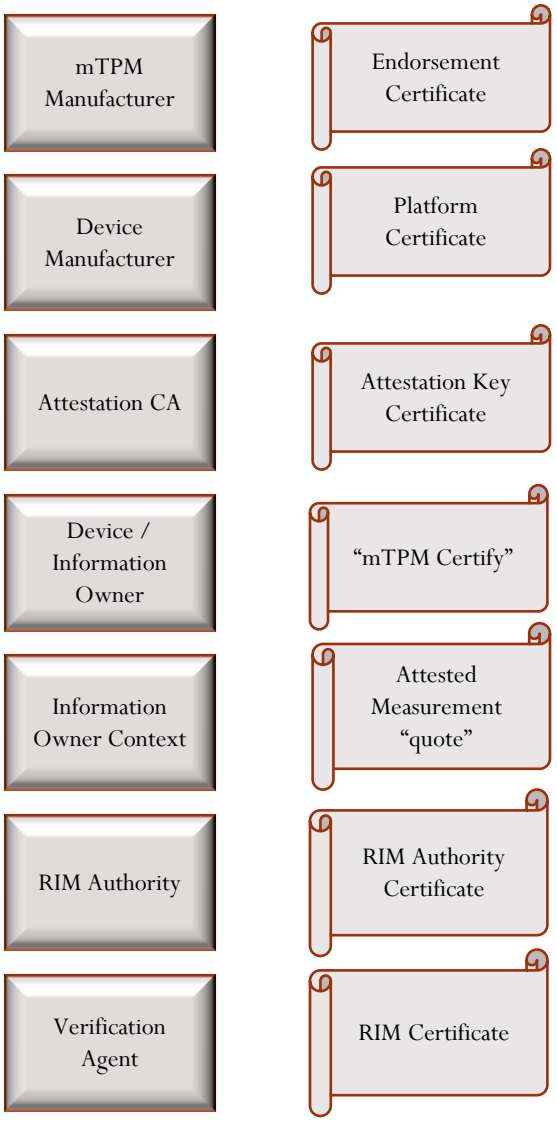
Transitive trust is a process whereby the RoTs establish the trustworthiness of an executable function, and trust in that function is then used to establish the trustworthiness of the next function.

Transitive trust may be accomplished either by: (1) knowing that a function enforces a trust policy before it allows a subsequent function to take control, or (2) using measurements of subsequent functions so that an independent evaluation may establish the trust. The mTPM may support either of these methods.

Code is signed so that the identity of the authority for the code is known. In modern architectures, where firmware and software components come from many different suppliers, it is often not feasible for platform manufacturers to know the signers of all code that runs on a platform. Therefore, they may not remain the authority on platform state for very long. The measurements recorded in the RTInt then determine the chain of authority for the current system state.

Attestation Hierarchy

Certificates provide assurances that the trust roots have been implemented in a way that renders them trustworthy



How Are Assertions Created?

- Verified components assert the state of peripherals, such as whether or not a microphone is live, or a camera is on. Verified components assert their state by hashing a certificate along with a fixed value representing the state of a peripheral or software element.
- When a device [or container] asserts its state with cryptographic primitives, it uses signing keys. These signing keys represent the tangible identity of some entity, whether it is the device, the device owner, the information owner, or an application on the device serving as a proxy for one of them.
- As the device and verified components create assertions, they must also create policies that spell out which information owners (as represented by their identities) have the right to share them.
  - As the device boots, it contains few information owners (maybe only one, that of the device owner). As assertions are created they can be stored along with policies that control access to them.
  - As new applications are provisioned and new information owner identities are created, the new applications may request access to existing assertions to which the device owner can grant or deny.
- When the applications are executed, the information owner identities are activated along with their assertion policies. If the running applications generate additional assertions on behalf of the information owner, then the information owner controls the policies of those assertions.
  - These assertions and policies are ephemeral in nature and should die when the application dies, and be recreated when the application runs again.
- Policy Manager collects assertions and determines whether or not the platform is in compliance with a particular policy, and whether requests to access assertions can be granted.

Policy Flexibility

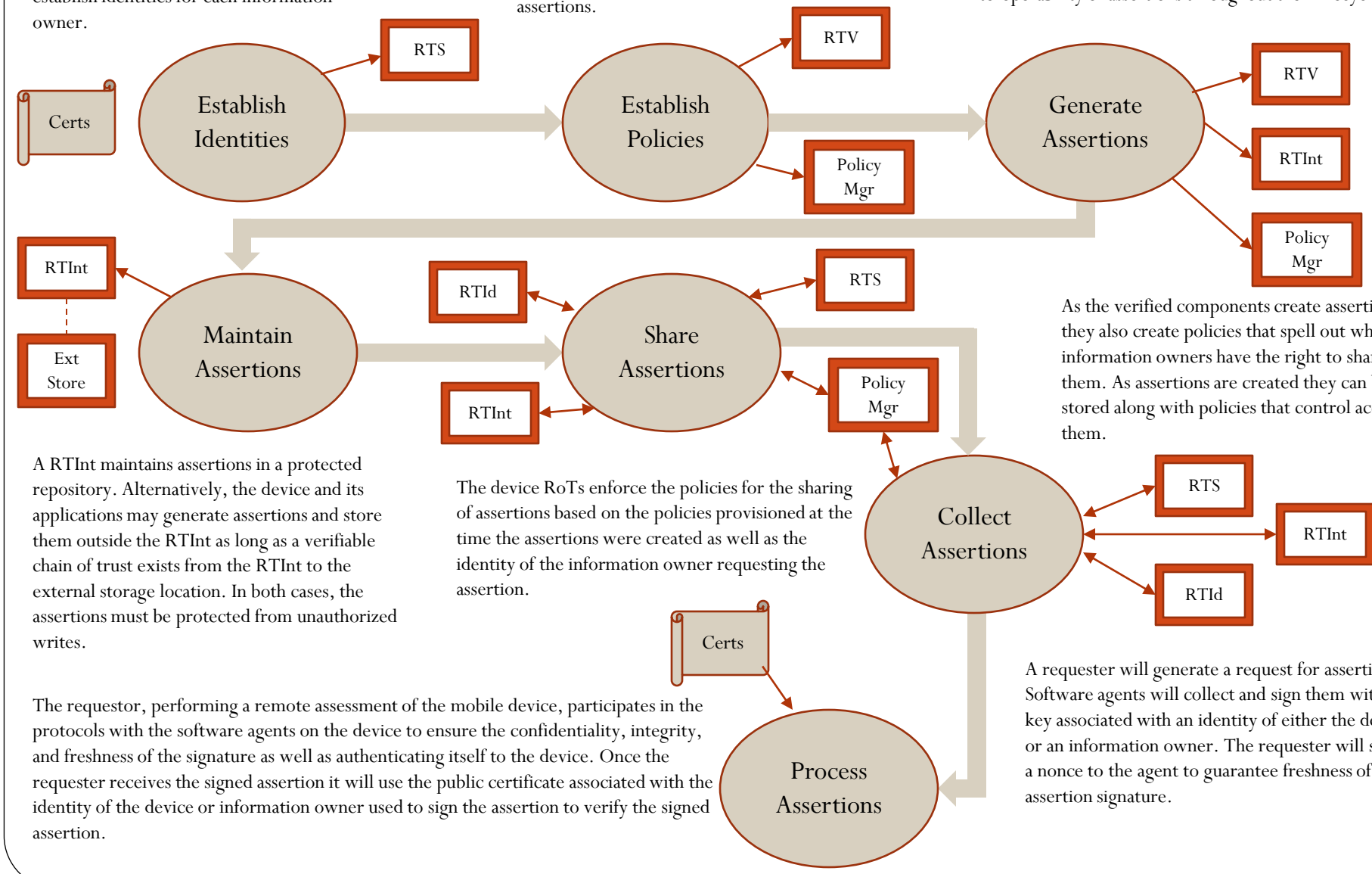
- The Policy Manager provides information owners with the ability to express the control they require over their information. The Policy Manager interacts with all of the information owners and translates the desired requirements for storing and sharing their information into the appropriate device and network configurations and policy.
- The Policy Manager must be trusted to implement the information owner's requirements correctly and to prevent one information owner's requirements from adversely affecting another's. The Policy Manager will enforce the policy with the strictest requirements. Where there is an irresolvable conflict the Policy Manager will notify the device owner and enforce a default policy that prevents unauthorized access to data until the conflict is resolved.
- In order to perform key functions, the Policy Manager must be able to query the device's configuration state. For example, if device sensor events are part of a policy to be enforced, the Policy Manager must have access to those outputs in order to enforce the policy, independent of the application context.

Assertion Life Cycle

Upon provisioning, a mobile device establishes identities for the device and the device owner. The owner should authorize the platform to establish identities for each information owner.

The device uses policy to control the information owner's access to assertions about the device and to establish explicit assertions.

Mobile devices generate assertions during boot by verifying the signatures of the boot components. It is important that standards be used to support interoperability of assertions throughout their lifecycle.



As the verified components create assertions, they also create policies that spell out which information owners have the right to share them. As assertions are created they can be stored along with policies that control access to them.

A RTInt maintains assertions in a protected repository. Alternatively, the device and its applications may generate assertions and store them outside the RTInt as long as a verifiable chain of trust exists from the RTInt to the external storage location. In both cases, the assertions must be protected from unauthorized writes.

The device RoTs enforce the policies for the sharing of assertions based on the policies provisioned at the time the assertions were created as well as the identity of the information owner requesting the assertion.

A requester will generate a request for assertions. Software agents will collect and sign them with a key associated with an identity of either the device or an information owner. The requester will supply a nonce to the agent to guarantee freshness of the assertion signature.

The requestor, performing a remote assessment of the mobile device, participates in the protocols with the software agents on the device to ensure the confidentiality, integrity, and freshness of the signature as well as authenticating itself to the device. Once the requester receives the signed assertion it will use the public certificate associated with the identity of the device or information owner used to sign the assertion to verify the signed assertion.

Places Where Assertions Are Needed

- Assertions generally around 3 key device capabilities: Device integrity, Isolation, Protected Storage
- To support typical device / container management objectives:
  - Securing mobile credentials (e.g. unique device information owner IDs, Bluetooth & WLAN IDs, etc.) to establish the authenticity of a device or information context on the device to store process, or access resources.
  - Attested updates over the air (e.g., FOTA, patches, policies) or remote diagnostics & auditing
  - Data backup and restore, remote lock and wipe, service enrollment / de-enrollment
  - Device compliance status reporting & subsequent service decisions. Enterprises check software state and OS policy compliance before allowing access to corporate network.
  - Secure storage by information owner context to provide evidence to information owners that the confidentiality and integrity of sensitive data on the device is protected while at rest, while in use, and upon revocation of access.
  - Compromise or disruption in processes running in other execution environments on the platform will not cause loss of integrity in the software processes used in the subject execution environment.
- To support typical device owner transactions:
  - To provide assurance to the device owner that they will not lose control over the device or overextend privileges to "guest" information owners. Overly broad enterprise policies can be detrimental to device owners - full device wipe, remote lock, reset device configuration by an information owner other than device owner
  - To provide confidence in transactional integrity properties to relying parties. A device may assert specific claims about operating status in a way that information owners can confidently rely upon to make decisions about interacting with the device, e.g., a secure PIN entry method was used for a mWallet transaction.
  - To avoid data spills when using multi-context applications and peripherals (e.g., browser, GPS, Bluetooth, WiFi). For example, when a browser residing in a Carrier domain connects to multiple information contexts (e.g., enterprise and banking).
  - To support migration of credentials from one device to another

Platform Assertions

- Identities
  - identity of the device
  - identity of the Device Owner
  - Identity of Information Owner
- System static images
  - Boot firmware (hash)
  - Kernel (hash)
  - Drivers (hashes)
  - Other key system files (hashes)
  - CA trust list
  - Digital signature of the kernel is valid
  - Digital signature of the drivers are valid
  - Digital signature of the key system files are valid
- System capabilities
  - Ability to arbitrate isolation requirements (system features that support isolation)
  - I/O features
  - Hardware (presence of peripherals)
- Current configuration state
  - debug state of device
  - Location
  - Sealed storage
  - Anti-malware protection active
  - Firewall configuration
  - Network state (WAN, WiFi, bluetooth, NFC, etc)

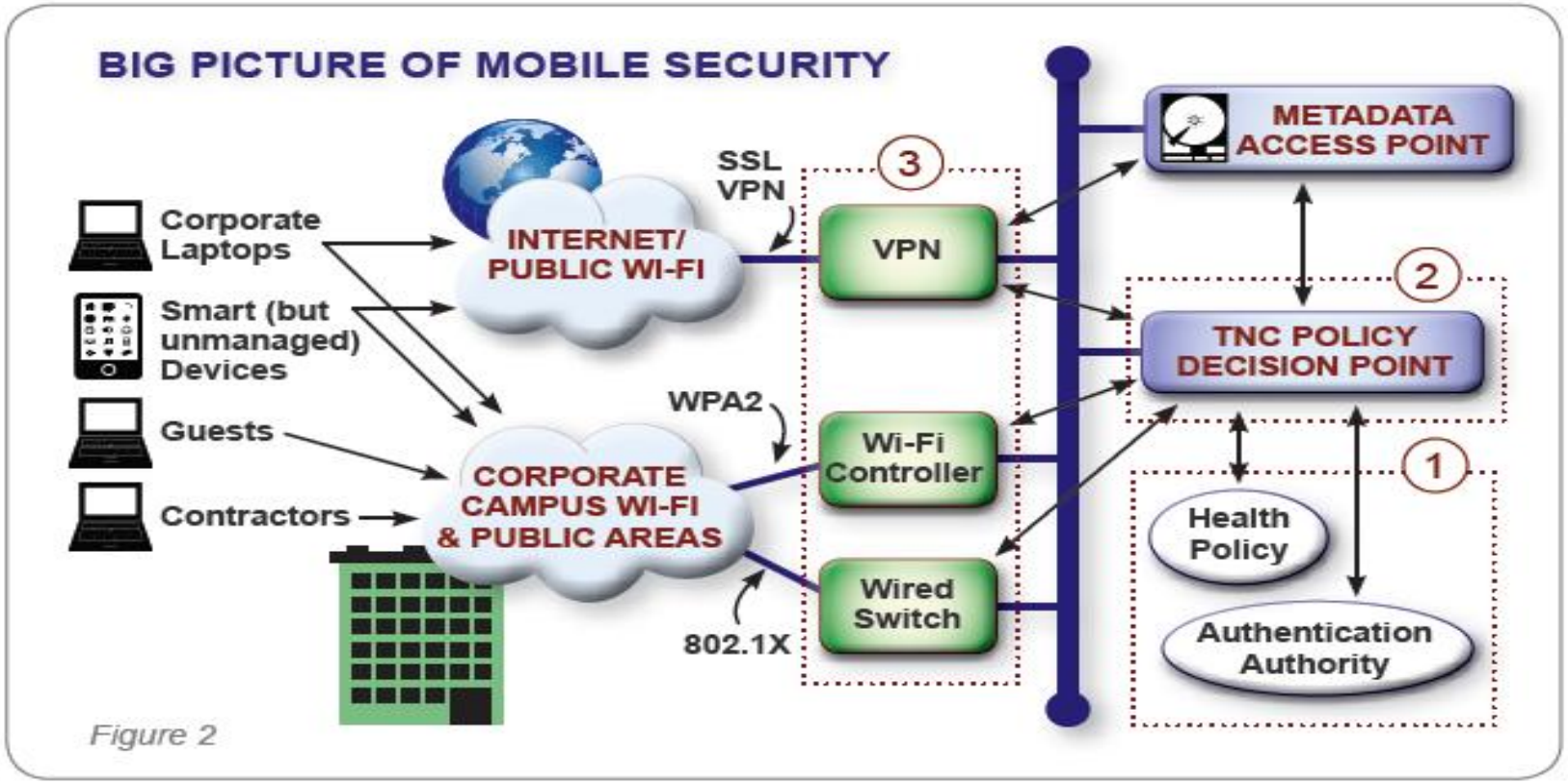
Transaction Assertions

- Proof of possession of an authoritative token is used as the basis to begin a trusted interaction – trusted credentials or attributes
  - "Trusted credentials" must have integrity and strong identity assertion; starting (ideally) with some authoritative, immutable, tamper-resistant source, e.g., smart card, mTPM
- Authoritative token / credentials are bound to respective device-user-transaction
  - Bindings should flow end-to-end in both directions
  - Bindings provide basis for authentication, authorization, accountability
  - Assertions are signed based on keys associated with the device/user/information owner identity.
  - Non-repudiation of the transaction, protection against replay attack are types of assertions enabled.
- Trusted assertion reporting
  - Need 2-way assertion flow
  - Need to allow verification on the device or in the cloud
  - The consumer device asserts the integrity of the information provided, the authenticity of the user providing the information, and (optionally) the integrity and authenticity of the execution environment that enabled the other assertions.

Inter-Context Assertions - When assertions must be generated for access across different information owner contexts

- When the assertions required straddle the boundary between Information Owner context owners, it becomes critical to understand the limits of the *isolation mechanisms* built into the device.
- Other Key Assertions:
  - Sealed Storage - Execution of trusted processes and access to data is based on a known good (measured) state of the overall platform or of the specific isolated execution environment that manages the information owner context and related data. Access to trusted engines is sealed based on mTPM state / measurements.
  - Locked Storage - Upon power-up, Information Contexts and their associated data should originate in a locked / encrypted state. Transition from the locked / encrypted state to the unlocked/decrypted state occurs if the Information Owner provides the correct authentication, and the RTS / RTInt are satisfied of the state conditions.
  - Trusted Path – e.g., an application has exclusive access to path to radio and UI. Requires assertions regarding the authenticity of components talking and being talked to; and the secrecy of data being transmitted.
  - Trusted API - There must be assertions regarding controlled interfaces between contexts
- Issues with Inter-Context Assertions
  - SSO to different containers
  - What if one container hibernates?

End-to-End Assertions



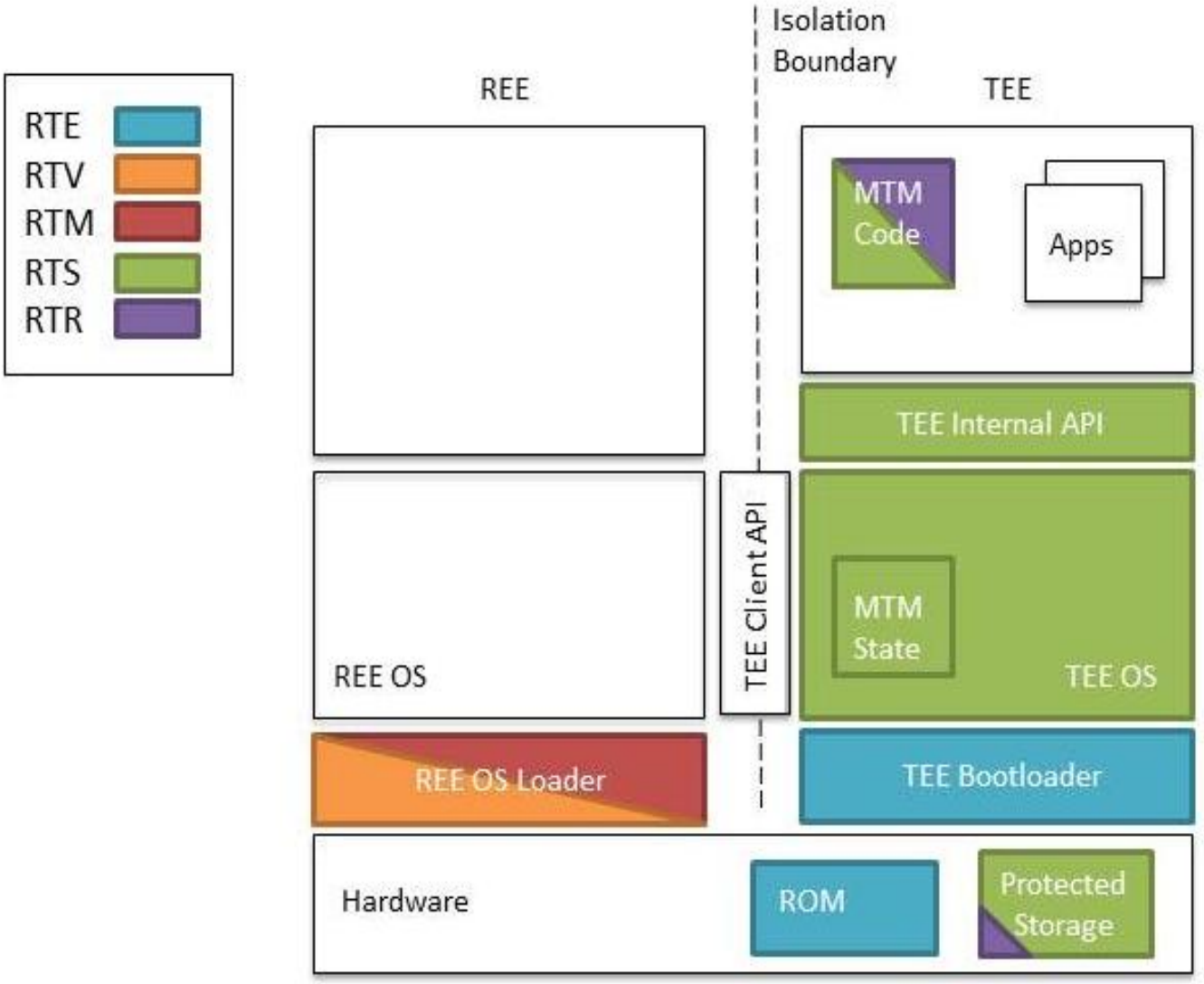
- A TNC-centric view of secure mobility: alternatively, a smart phones can attest directly to a server via the public internet i.e. without going via TNC

End-to-End Assertions Using TNC

- Assessment Options
  - Identity, health, and/or behavior
  - Optional hardware-based assessment with mTPM
  - Pre-admission, post-admission, or both
- Enforcement Options
  - 802.1X, firewalls, VPN gateways, DHCP, host software
- Clientless endpoints
  - No NAC capabilities built in
  - Printers, phones, robots, guest laptops
- Information sharing
  - IF-MAP lets security devices share info on user identity, endpoint health, behavior, etc.
  - Federated TNC supports federated environments

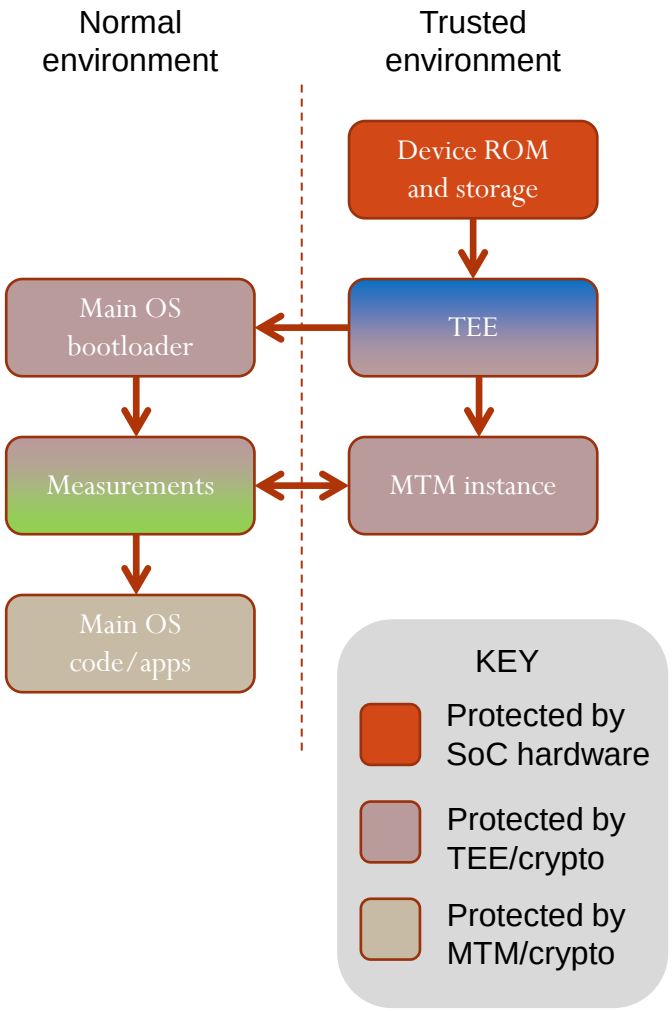
Issues with BYOD – Integrated or Co-Existing Trust Engines

- Examples of trust engines include Global Platform TEE, hypervisor, SMM, mTPM (MLTM / MRTM), etc.
- How to allocate / realize RoTs?
- How to evaluate robustness / compounding of assertions, i.e., some from TEE, some from hypervisor, etc.?
- How to certify?
- How to ensure isolation among trust components within same TEE for any shared resources, e.g. RTS exclusiveness with respect to the mTPM component?

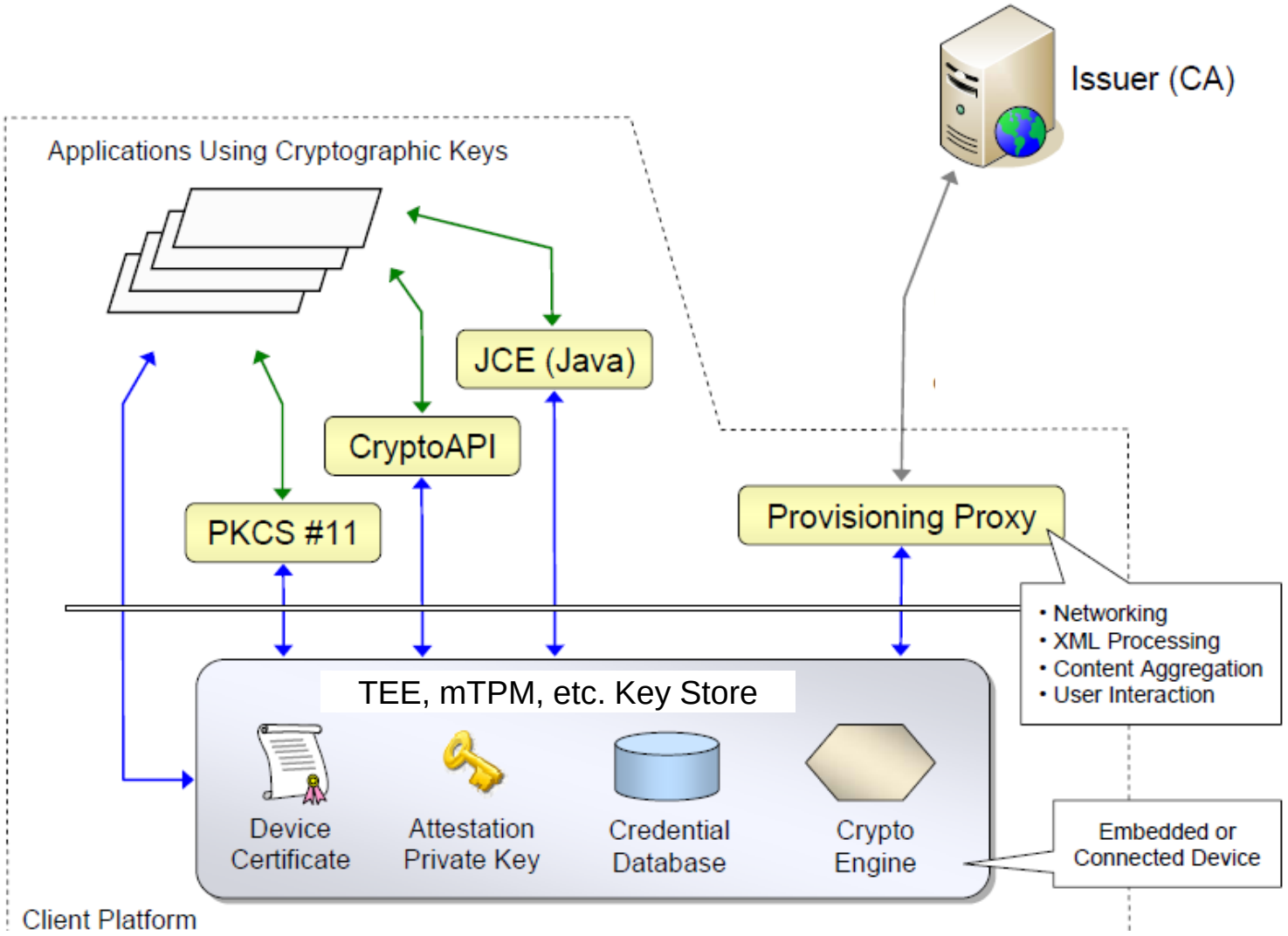


Notional Integration of GP TEE and mTPM

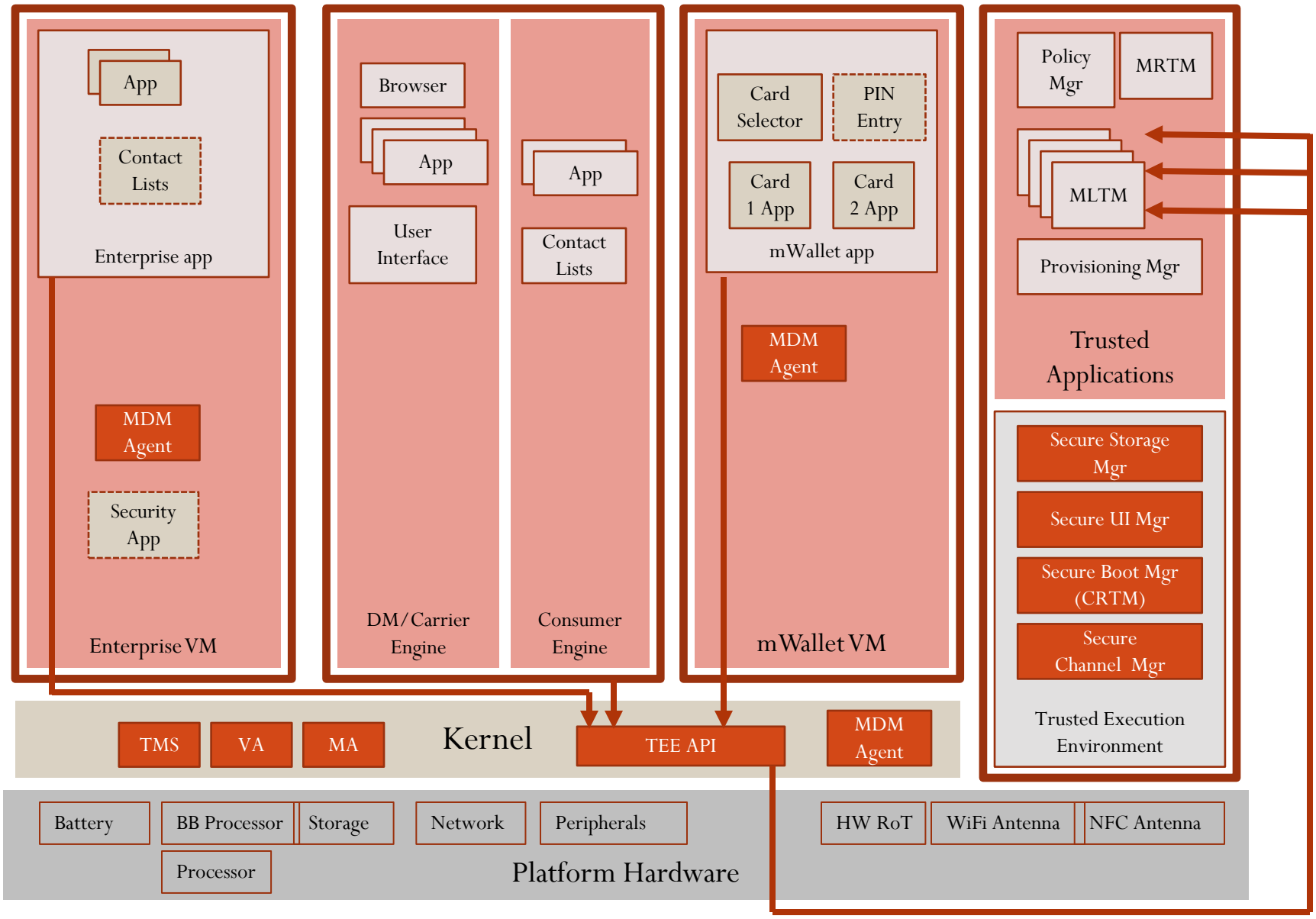
1. TEE Supervisor application will create a new guest VM without any associated resources.
2. Upon success, an in-kernel object representing the new guest VM and its state is created.
3. The guest VM created during the preceding step does not yet have any resources associated with it.
4. It is the responsibility of the TEE supervisor application to allocate resources for the guest VM (e.g., mTPM). It is not mandatory to allocate all resources used by the guest VM before the first switch to non-secure world (normal environment).
5. Any resource allocation is checked by the in-kernel component of the TEE framework.
6. If multiple TEE supervisor applications and multiple guest VMs are running concurrently, the in-kernel resource manager takes care of prohibiting any conflicting allocations.
7. Normal World RTM generates measurements and assertions – stores in mTPM. Continues process and loads Normal environment OS.



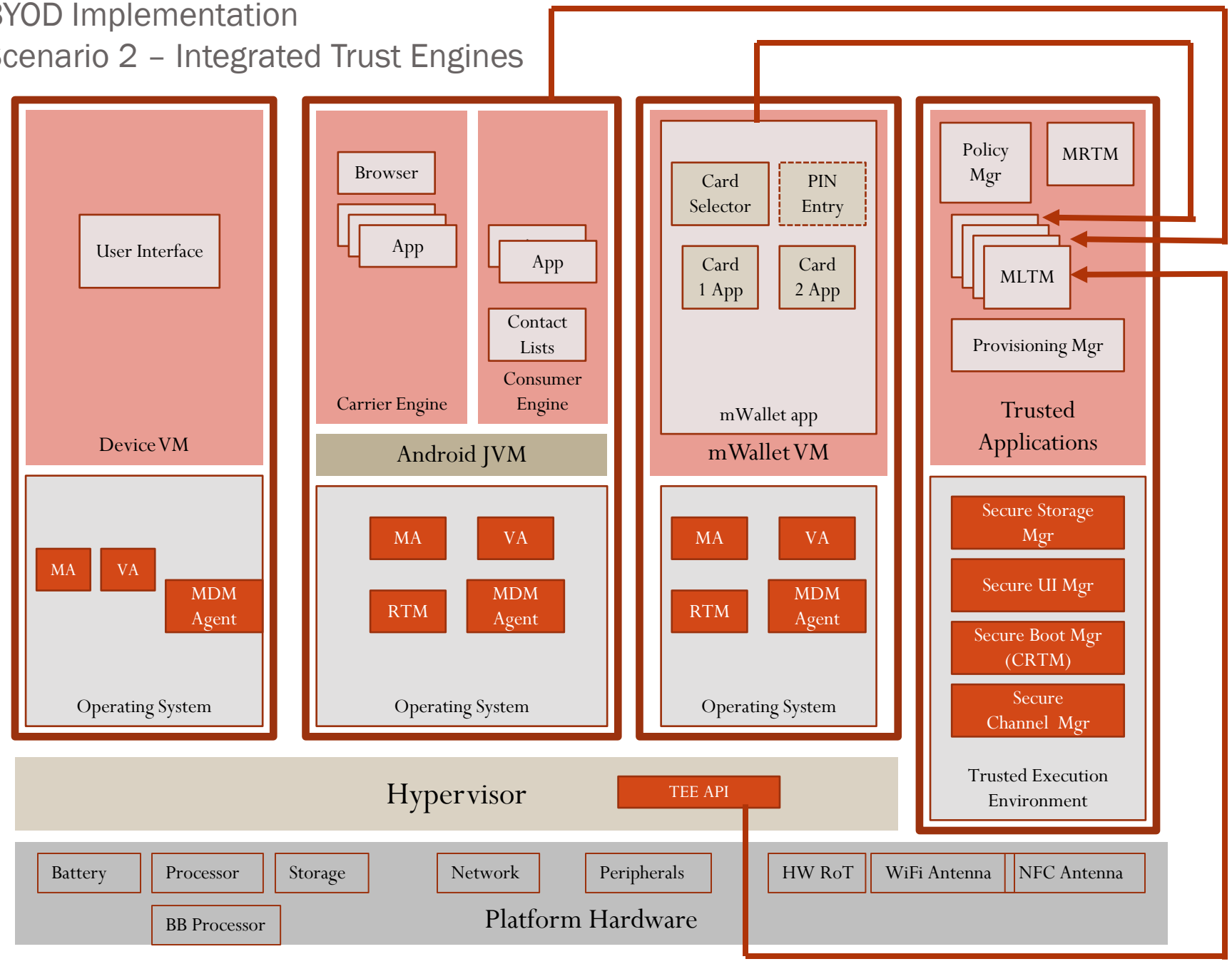
Issues with BYOD – Trusted APIs for Provisioning



BYOD Implementation Scenario 1 – Basic Case



BYOD Implementation
Scenario 2 – Integrated Trust Engines



BYOD Implementation Scenario 3 – Co-Existing Trust Engines