On the Secure Distribution of Vendor-Specific Keys in Deployment Scenarios

Nicolai Kuntze, Carsten Rudolph

On the secure distribution of vendor-specific keys in deployment scenarios

Nicolai Kuntze and Carsten Rudolph

Fraunhofer SIT (Germany)
{kuntze|rudolphc}@sit.fraunhofer.de

Abstract. Product counterfeit is a tremendous challenge for vendors in many areas. Particularly important is a prevention of product counterfeit where products like telecommunication devices interact with other systems and thus a malfunctioning of a single device can jeopardize the complete system. This can also deteriorate the reputation of the vendor. Furthermore, violation of intellectual properties can cause financial losses. Detection of product counterfeit can be based on tracking back each device to the production process of the vendor to ensure the product origin. Devices without a verified source can then be considered counterfeit with a high potential to be malicious or of low quality. Vendors already apply vendor-specific security technologies protecting the distribution. These often employ special hardware-based security mechanisms specifically designed for a particular range of products.
This publication shows the usage of the already available Trusted Platform Module to allow for distribution channel protection and to leverage overall security by allowing the secure identification of a specific device. It also explains a few additional Trusted Platform Module functionalities that can be used.

1 Introduction

Today's international collaboration and outsourcing of production processes results in a separation of device design and device production. One example is the production of network equipment such as routers or switches. The vendor controls the design but other companies (the manufacturers) are subcontracted to actually produce and assemble the devices. Thus, device blueprints and production is not under the direct control of the vendor, but controlled by the manufacturer. One core concern of the vendor is to be able to control that no additional devices are produced and sold bypassing the vendor. The manufacturer could in principle produce more devices and directly bring them to the market. Thus, the manufacturer would diminish the revenue of the vendor. Even worse, clients might believe they buy original products and request support and maintenance from the vendor. This creates additional costs.

Furthermore, outsourcing the production and other steps in the actual creation of the final product results in perils to the vendor's business and to the end users [1]. The manufacturerers have the knowledge and technology to also build

counterfeit products that cannot be easily distinguished from the original devices, but may have less quality, do not comply to the specification, or have back-doors and other malicious software installed. Counterfeit devices with low quality or malicious behaviour also cause a loss of reputation and endanger the brand experience if they cannot be reliably recognized.

The vendor is in a paradoxical situation. All information on how to build the device needs to be given to the manufacturer, while at the same time the vendor wants to keep control on the number and properties of devices being built and sold under the vendor's name. One possible approach to this catch-22 situation is to create individual registrations of each single device and track devices from manufacturer to the customer. The identification of each single device can be based on individual vendor-specific keys to be inserted into device or on a subsequent registration process using a vendor key not known to the manufacturer. Such a key needs to be protected against duplication. One aspect hereby is that a vendor specific key is only issued to devices after production and final tests to restrict keys to devices being sold. However this would require each device to be inspected by the vendor in a controlled environment, which adds a lot of additional costs.

The state of the art is based on vendor-specific or commercial off-the-shelf hardware tokens providing random-number generation and asymmetric cryptography. These tokens are developed by security hardware vendors and included by the contracted manufacturer into the devices. Tokens can be pre-equipped with unique keys, keys can be generated within the token or the vendor can directly inject a device identity into this hardware token. Through this process each device can be uniquely identified and the vendor can track these devices without requiring direct access to the complete final device in a controlled environment.

By controlling the hardware tokens, the vendor also keeps control over the number and identity of devices being shipped or getting into the market. This approach can provide some protection against counterfeit products, but it also has some disadvantages. First, an additional piece of hardware needs to be included that has no other benefit in the product. These hardware tokens are in many cases single purpose chips and in comparison to a multi-purpose commercial off the shelf chip relatively expensive due to smaller production numbers and additional overhead for logistics. Further, the process for establishing keys creates additional overhead, because the used security hardware does not support protocols suitable for a zero-touch registration and initialization. Another option is to use radio frequency identification (RFID) tags to create unique device identities (see [2]). The drawback with a RFID based solution is that any kind of (remote) enforcement is not applicable and that it requires RFID readers at various places in the value creation chain. It might be more efficient to use general purpose security hardware in the devices that also provides more generic security functionality useful for all other parts of the device's life-cycle.

Security industry has developed various approaches for hardware security anchors. Examples include TrustZone, SmartCards or the Trusted Platform Module (TPM) as specified by the TCG. The value of TrustZone strongly depends

on additional security functions provided by the platform, while SmartCards are usually not integrated into a platform but considered to be removable devices suitable e.g. for user authentication or digital signatures. In contrast to this, the Trusted Platform Module is a self-contained chip integrated into the device which offers an interesting basis to implement protection schemes suitable for industry. It provides a hardware-protected identity, various security functionalities and the possibility to establish device-specific keys that cannot be extracted or migrated to other devices. The goal of this paper is to present an approach to use one type of such generic commercial of the shelf security hardware, namely the TPM, to satisfy the industry demand for IP protection and control on logistic processes. Further, this publication will show additional benefits when using the TPM for authentication-only approaches.

This paper presents a concept for the establishment of vendor specific cryptographic keys using the Trusted Platform Module. In the remainder of the paper a more detailed scenario and first requirements towards the distribution process are defined in Section 2. Section 3 presents necessary basic concepts of the TPM for the context of this use case. Based on the requirements and TPM concepts a solution architecture is detailed in Section 4, followed by an analysis in Section 5. The publication concludes in Section 6 showing advanced uses for strengthening the customer's experience as an additional use of the security functionality.

2 Scenario

Given the setting of a vendor (i.e. the equipment manufacturer) outsourcing production to the actual manufacturer (i.e. the assembler) as described above, this section provides a structured and more detailed analysis of the addressed scenario. To counter product counterfeit an understanding of the involved stake-holders and their interactions in the course of the production of equipment is required.

Stake-holders The value creation chain of a product involves mainly Original Equipment Manufacturer (*OEM*), *Device Assembler*, *Distribution Partner* and *End User*. The role of the *OEM* differs in various business cases which impacts the counterfeit techniques suitable. In more detail, the workflow of producing a device can involve the following stake-holders:

OEM represents the publicly know vendor of the device. The *OEM* often creates the design, owns the intellectual property, and provides product liability and device support to the *End User*. Actual production of the equipment is very often outsourced to contractors to allow the *OEM* to concentrate on development and marketing and to reduce production costs.

Device Assembler represents the contractor of the *OEM* that produces and/or assembles the hardware device. This may actually be several companies not even in the same country (and not under the same jurisdiction) as the *OEM*.

Distribution Partner receives the devices directly from the *Device Assembler* in the name of the *OEM* and sells the device to the *End User*. Devices in stock are not handled by the *OEM* but by subcontractors.

End User buys and uses the device which is perceived as a product of *OEM*.

In addition to these stake-holders, the secure distribution of devices introduces additional security-critical stake-holders to the described scenario. For the proposed solution, the *TPM Supplier* is a vendor that produces and sells TPM hardware modules. The TPM can come with a security certification and pre-established key so-called Endorsement Key that also should be certified by the *TPM Supplier*. Given the security critical focus of these hardware chips, these vendors need to be trustworthy to the *OEM* and usually already have implemented secure production arrangements.

Functional Production Workflow Ignoring the steps for anti-counterfeit solutions, the following workflow describes the production of a device from a functional perspective. The *OEM* designs a new device using its resources and intellectual property. Then the *OEM* contracts a *Device Assembler* to actually build a larger number of devices. This includes committing the blueprints of the new design and giving all information necessary to build the device to the *Device Assembler*. The *Device Assembler* produces devices according to the blueprints of the *OEM*'s design and might provide evidence, test results, production logs, etc. to the *OEM*. The devices are delivered from the *Device Assembler*'s site directly to the *Distribution Partner*. The *Distribution Partner* sell and ship the final device to the *End User*who then can register the device with the *OEM*.

Security Analysis Given the above work flow, obviously, by contracting of the *Device Assembler*, the *OEM* loses control of the production process. In this step, the *OEM* provides the *Device Assembler* with all blueprints necessary to produce the device. After this step, the *OEM* has no direct control over the amount of devices produced or sold. It completely relies upon the *Device Assembler*'s and/or *Distribution Partner*' accounting of devices shipped. The *Device Assembler* possibly colluding with one of the *Distribution Partner* has various options to cheat in this process.

Threats The *Device Assembler* possesses the required blueprints and capabilities to manufacture devices. It might sell these to counterfeit producers or itself produce and sell more devices than it accounts for with the *OEM*. An involvement of *Distribution Partner* and the requirement for call for bids further increase potential for counterfeits purchases by *End User*. Neither *End User* nor *OEM* would be able to reliably recognize counterfeits (beyond potential alterations in production blueprints). Thus, several types of malicious actions by the *Device Assembler* can be distinguished. (i) *Device Assembler* builds a higher number of devices in accordance with the blueprint of the *OEM*. These devices cannot be distinguished from original devices. The only difference is, that the *OEM* does not get any revenue from these devices and the *Device Assembler* gets a much bigger share. Clearly, the *OEM* loses revenue and at the same time is responsible for maintenance, warranty, etc. (ii) *Device Assembler* builds a higher number of the same devices, but sells them under a different brand name, stealing the *OEM*'s IPR. (iii) *Device Assembler* builds devices that are of lower quality,

use cheaper hardware elements or have other (possibly intentionally malicious) changes to increase its revenue. This decreases the *OEM*'s revenue and also creates additional risks for the *End User* using the devices.

Security Requirements The threats described above yield the following security requirements. (i) The *OEM* needs to be able to identify non-counterfeit products and to track all devices sold with the *OEM*'s brand. (ii) Counterfeit devices must be distinguishable to the *OEM* and (depending on the type of device), remote detection of deployed counterfeit products can be necessary. (iii) *End User* need to be able to distinguish counterfeits from original products. This might require the help of the *OEM*. (iv) The *End User* should to be able to recognize counterfeits using an offline scheme without active work of the *OEM*.

3 Trusted Computing Basics

According to the mission statement of the Trusted Computing Group (TCG), Trusted Computing based on hardware roots of trust has been developed by industry to protect computing infrastructure and end points. The TPM provides the core security functions and serves as a root of trust for each individual device.

TCG defines the currently used and market available TPM in version 1.2 but also provides a specification for the next generation of TPMs in form of the version 2.0. Both versions of the TPM design share underlying principles regarding the functionality and functions provided. The TPM is regarded as a trust anchor bound to an individual system. Trust is defined within the TCG to convey an expectation of behaviour. Hereby it needs to be emphasized that a predictable behaviour does not constitute behaviour that is secure or worth to be trusted. However, various security properties for platforms can be satisfied and controlled based on the TPM. For example, to determine the trust in a certain platform, it is required to identify the identity of the platform. The TPM provides a unique identity for a platform which can either used to directly identify a specific platform or provide for pseudonymous identification. The next sections give a more detailed presentation of how a TPM is used to implement its dedicated role as a root of trust. Specifically, it is shown how different roots of trust in a system design complement each other to build platforms with particular security properties. A more detailed description and also explanation of the TPM commands used can be found in the book by Chris Mitchell [3].

3.1 Roots of Trust

The high level concept of Trusted Computing as defined by the TCG introduces different roots of trust in the system design providing complementary security functions (see also [4]). To attest on the health of a system each software component needs to be measured beginning from the initialization of the device. In the reference design according to TCG, the initial start of a system begins with the **Root of Trust for Measurement (RTM)**. A RTM measures itself and is implemented using platform features to ensure the tamper resistance of

the RTM. The RTM measures the next stage in the boot process and transfers control to it. By this, the RTM already behaves according to the *measure before execute approach* underpinning the overall measurement of the boot process and of software subsequently started on the system. The second component is the **Root of Trust for Storage (RTS)**, which is usually implemented by protected non-volatile storage within the TPM hardware. The RTS mainly has three roles. First it needs to provide secure storage for the cryptographic identity of the TPM. Second, it also provides secure storage for the keys that are used to encrypt data to be securely stored on (insecure) storage media on the platform or outside the platform (e.g. symmetric keys for bulk encryption). Finally, the third role includes particular protected registers used to store the measurement information as hash-values (i.e. chains of hash values represented by the final value in the chain.). These registers are called Platform Configuration Registers (PCR). The final root of trust is a securely stored cryptographic key (Endorsement Key EK), the **Root of Trust for Reporting (RTR)**. This key is used to create so called Attestation Identity Keys (AIKs) that are then used to digitally sign PCR content in a TPM_Quote and to certify non-migrateable keys generated by the TPM. AIKs can be used to identify a platform with different levels of pseudonymity [5].

3.2 Basic Trusted Platform Features

The TPM provides a large set of security functionalities. The secure distribution of vendor-specific keys in deployment scenarios builds on and uses authentication, attestation and protected location functionalities. **Authentication** can be easily based on the Endorsement Key (EK) that is either established in the TPM by the producer of the TPM or implanted to the TPM in a later stage of the production process. Together with a certificate for the EK, this key is used to build subsequent authentication processes. **Attestation** in this context describes the process of reliably reporting the platform status. The TPM provides the functionality for secure remote attestation. Thus, remote entities can get digitally signed information on the current content of PCRs. Protocols for remote attestation are defined in the TCG standards. **Protected Location** for keys and other data transferred to the platform is provided by the TPM. In the TPM context, the process of encrypting data with a key protected by the TPM and binding this encrypted data to a particular state of the platform (i.e. particular PCR values) is called *Sealing*. Thus, sealed data can only be decrypted when two conditions are satisfied. First, the same TPM needs to be used with the correct key loaded to the TPM (and optional the correct authentication value/password for the key is given) and second, the platform is in the correct state that the PCR values match.

4 Vendor specific key establishment

As described in the previous section, TPMs provide a different kind of unique identities. These identities allow to establish secure channels to the TPMs or

respectively a given device. These unique identities are implemented as cryptographic keys that are protected by hardware inside the TPMs. This ensures a higher protection level compared to solely software based solutions.

Based on the concepts of the TPM, there are several ways to establish vendor and end user specific identification keys. These keys may be generated externally and injected into the TPM, where they are stored securely, or the TPM's internal Random Number Generator and Key Generator can be used to create keys and their binding to a certain TPM can be validated. Both of these schemes are presented in this section using different characteristics. The logistics processes involved with each of these approaches are also outlined.

4.1 TPM Generated versus Vendor Injected Keys

The usage of the Endorsement Key is restricted to only very few operations. In particular, it cannot be used for digital signatures at all and for decryption only during a very constrained set of operations. These restrictions are in place, in order to use the TPM in scenarios (such as Consumer Electronics) where a unique identification of devices is not desired. However, TPMs provide additional means to establish context related identities that can be used for the given scenario of counterfeit protection. Generally, vendor established keys can be based on either of two approaches – Generated externally and injected into a device / TPM or generated by the TPM and recorded / certified in a secure way.

Key Injection In the case of key injection, the *OEM* or *End User* create keys to be used as (context related) identification externally and inject them into the TPM. After the key injection, the keys are protected by the TPM and can be utilized as a primary identity credential for a given device. Based on these keys, the device can proof its origin and identity. This can also be utilized in other technical as well as organizational processes. To ease the process, the *OEM* or *End User* may issue certificates for these injected keys.

TPM created Keys The TPM has the ability to create keys on its own, using a strong random number generator. These keys are then protected by the TPM chip, such that the private portion is never known or usable outside of the specific TPM that generated it. In order to achieve an authentication against the *OEM* or *End User*, the public portions of these keys may be read out in a protected environment and stored in a database or the *OEM* or *End User* may issue corresponding certificates. Furthermore, it is possible to certify the origin of these keys using the TPM's functionalities themselves based upon previously established identity keys. An establishment of TPM generated keys is even possible remotely without a protected environment or channel, given that an identity relation has been established before using another key.

4.2 Local Establishment of a TPM 1.2

The workflow for the local establishment of vendor specific key material with a certain TPM works similarly to the deployment of custom security tokens. The

OEM or *TPM Supplier* generates key material and injects it into the security chip (in this case the TPM) in a protected environment. The public portion is stored for identification and the private portion is usually deleted, such that it only exists inside the TPM.

Logistics The logistics process of physical objects (focused on the TPM chip) in this approach acts are as follows. (i) The *TPM Supplier* produces a TPM (optionally without an Endorsement Key (EK) and EK certificate). (ii) The TPM chip is delivered to a protected environment at the *TPM Supplier* or the *OEM* facilities. (iii) Local, in the protected environment, keys are created by or injected into the TPM and their public portions recorded in the *OEM*'s database. (iv) The TPM chip is delivered to the *Device Assembler*. (v) The TPM chip is assembled into the target device. (vi) The target device is shipped and distributed. (vii) Remotely, at any time after assembly (from testing at the *Device Assembler* to the *End User*), additional keys can be created by the TPM or injected into the TPM using the previously established keys.

Identity Establishment In this usage scenario an establishment of the TPM provided identity means is not necessary, since it is used in a protected environment. The logistics process and trustworthiness between the *TPM Supplier* and the *OEM* need to ensure that a specification conform TPM is used. The TPM needs to be owned (using TPM_TakeOwnership) in order to enable its Root of Trust for Storage, which also activates the ownership of the EK for identity reporting. The *End User* may change the credentials for identity usage after deployment without invalidating the keys. The only constraint is that the ownership may not be cleared, since this would invalidate the vendor supplied keys.

Local Injection of Key Whilst the TPM is in the protected environment at the *TPM Supplier* or *OEM* facilities, it is possible to inject vendor specific keys (for storage, signing and binding; not for identity) into the TPM key hierarchy. To do so, the *TPM Supplier* or *OEM* generate a new key-pair and construct a MigrationBlob that is targeted at the TPM's Storage Root Key (SRK) using TPM_ConvertMigrationBlob for importing. In order to do so, the TPM must have been taken into ownership before (the SRK's authorization value can be set to the "well-known secret" as defined by the TCG). This process can be repeated several times. Also, if one of the newly created keys is a storage key, this may be used for further key injections as parent. The public portions of these keys (or their fingerprint) are securely saved in the *OEM*'s database for established keys, such that they can be validated in the future. Alternatively, the *TPM Supplier* or *OEM* may issue according certificates to be shipped with the TPM and later stored on the assembled device. The private portions can be deleted, as they are wrapped in TPM bound key blobs. These wrapped key blobs must be shipped with the TPM to the *Device Assembler* to be accessible inside the assembled device (the (limited) TPM non-volatile storage area can be used for this).

Local TPM-based Key Creation Whilst the TPM is in the protected environment, it is possible to let the TPM generate new keys using its internal Random Number

Generator (using TPM_CreateWrapKey or TPM_MakeIdentity) for use as vendor specific keys (for storage, signing, binding as well as identity and sealing). To do so, the TPM must have been taken into ownership before. The SRK is one of these TPM generated keys and can be used as vendor specific storage key itself (without having to be stored in an additional wrap key blob). The public portions of these TPM generated keys can be read (from the public portion of the KeyStorageBlobs) and must be stored in the *OEM*'s database for established keys or an according certificate must be shipped. The private portions are only known to this TPM and are wrapped in TPM bound key blobs. These wrapped key blobs must be shipped with the TPM to the *Device Assembler* (except for the SRK; the TPM's NV storage area can be used for the other keys).

Remote TPM-based Key Generation or Injection after Assembly If one of the keys that were created in the protected environment is a signing key, it is possible to remotely create new keys and use these vendor specific signing keys to certify (using TPM_CertifyKey) that the newly created keys were generated inside the same TPM and the signing key. This is similar to the Remote Establishment of a TPM 1.2. If one of the keys that were created in the protected environment is a storage key, it is possible to remotely inject new keys and use the vendor specific storage key as target for a MigrationBlob containing this newly created key via the TPM_ConvertMigrationBlob. (This is similar to the Remote Establishment of a TPM 1.2) Both of these processes can happen while the device is at the *Device Assembler*, *Distribution Partner* or even *End User*.

4.3 Remote Establishment of a TPM 1.2

The workflow for the Remote Establishment of vendor specific key material in a TPM 1.2 does not require the existence of a protected environment under the control of the *OEM*. Instead, TPM chips are shipped directly to the *Device Assembler* and the key material is established after device assembly (potentially during a testing and registration phase). It does however require and active network session from the newly assembled device to the *OEM*.

Logistics The logistics process of physical objects (focused on the TPM chip) in this approach acts are as follows. (i) The *TPM Supplier* produces a TPM (typically shipped with an Endorsement Key (EK) and EK certificate). (ii) The TPM chip is directly delivered to the *Device Assembler*. (iii) The TPM chip is assembled into the target device. (iv) Remotely, keys are created by or injected into the TPM by the *OEM* and their public portions recorded in the *OEM*'s database. (v) The target device is shipped and distributed. (vi) Remotely, at any time after assembly (from testing at the *Device Assembler* to the *End User*), additional keys can be created by the TPM or injected into the TPM using the previously established keys.

Identity Establishment The approach requires an establishment of a (pseudonym) identity inside the TPM. This can be done with standard TPMs that provide

an EK certificate via several approaches (such as [6]) or the *OEM* may have requested certain custom identity establishment from the *TPM Supplier*. This identity only needs to account that it actually belongs to a TPM and is free of collisions. The counting of TPM identities is used to cross check the *Device Assembler*'s production accountings. As a result of this step, there exists an Attestation Identity Key with the TPM that can be used for further operations.

TPM-based Key Creation It is now possible to remotely use the TPM to generate key pairs (via TPM_CreateWrapKey and TPM_MakeIdentity) using the TPM's internal Random Number Generator (for storage, signing, binding, identity and sealing). Then the Identity Key can be used to certify (via TPM_CertifyKey) that the newly generated key was actually generated by the same TPM that the Identity Key is bound to. From the certificate, the *OEM* may store the public portion (or the fingerprint) of the certified key in the database for established vendor keys or issue a certificate to be shipped with the device. This process may also be performed again after sales of the device at the *End User*, even with newly created identity keys instead of the previously established Identity Key. The original Identity Key can even be deleted.

Injection of Key It is also possible to remotely inject keys generated by the *OEM* (for storage, signing and binding; not for identity). To do so, the *OEM* generates a key pair and creates a MigrationBlob using one of the certified TPM residing key. This may also include using the Storage Root Key directly. These MigrationBlobs can be imported into the TPM key hierarchy using TPM_ConvertMigrationsBlob and serve as additional vendor specific keys. The *OEM* would save the public portions of these key pairs in its database and delete the private portions, that would be saved in wrapped key blobs on the device. This process may also be performed again after sales of the device at the *End User*, even with newly created identity keys (must be TPM-based Created) instead of the previously established Identity Key. The original Identity Key can even be deleted.

4.4 Local Establishment of a TPM 2.0

With the specification of TPM 2.0 an additional key hierarchy has been introduced that can be used by *OEM*s without interfering with *End User*' use of the TPM. This key hierarchy is called platform hierarchy. With the *OEM* being the designer and administrator of the platform, this key hierarchy is exclusively used by him. The presented approach again requires a protected environment for secure establishment of some keys. Additional keys may be introduced in the future. We do not provide a remote establishment here as the necessary identity establishment is out of scope for this contribution.

Logistics The logistics process of physical objects (focused on the TPM chip) in this approach acts are as follows. (i) The *TPM Supplier* produces a TPM 2.0 (optionally without an Endorsement Key (EK) and EK certificate). (ii) The TPM chip is delivered to a protected environment at the *TPM Supplier* or the

OEM facilities. (iii) Local, in the protected environment, a Platform Hierarchy primary key is generated in the TPM the public portion recorded in the *OEM*'s database. (iv) Local, in the protected environment, any number of additional keys in the Platform Hierarchy can be created and their public portions recorded in the *OEM*'s database. (v) The TPM chip is delivered to the *Device Assembler*. (vi) The TPM chip is assembled into the target device. (vii) The target device is shipped and distributed. (viii) Remotely, at any time after assembly (from testing at the *Device Assembler* to the *End User*), additional keys can be created by the TPM or injected into the TPM using the previously established keys.

Identity Establishment In this scenario, an establishment of TPM identity is not necessary, since it is operated in a protected environment. In the protected environment at the site of the *TPM Supplier* or *OEM* the primary key for the Platform Hierarchy is created using TPM2_CreatePrimary. This key serves as a root key for this hierarchy and protects all subsequently created *OEM* keys. The *OEM* reads and stored the public portion of this key in its database via TPM2_ReadPublic. This can be used for the remote key creation in the future.

Injection of Key Under the Platform Hierarchy the *OEM* can import any number of keys for different purposes using TPM2_LoadExternal and TPM2_Import. In contrast to TPM 1.2 it is possible to also assign advanced usage policies to these keys. It is now possible to import symmetric keys into the TPM.

TPM-based Key Creation Under the Platform Hierarchy, the *OEM* can create any number of keys for different purposes using TPM2_Create. It can then read out the public keys inside the TPM and store it in the database for vendor keys.

Injection of Keys remotely Using TPM2_LoadExternal and TPM2_Import of the TPM 2.0, it is possible to encrypt a new key with an established key. This way, it is possible to ensure that only the TPM that was targeted can decrypt and use a newly generated key. This allows the *OEM* to inject additional keys into the TPM even after assembly, when the TPM is not in the protected environment.

TPM-based Key Creation remotely The TPMs Key Generation capabilities can be used remotelyusing TPM2_Create. The TPM is instructed to generate a new key under the Platform Hierarchy. Then an established storage key is used with the concept of TPM2_ActivateCredential to attest that the newly created key belongs to the same TPM as the previously established key.

5 Analysis

The *OEM* now has the ability to establish vendor specific keys with its devices that can be used for many applications. This section revisits the requirements and analyses their fulfilment. Further, it discusses additional functionalities of TPMs that can be utilized and analyses potential residual threats in this approach.

5.1 Requirement Fulfillment

Given the requirements from Section 2 we come to the following analysis of requirement fulfilment. First, the *OEM* needs to be able to identify non-counterfeit products and can track all devices sold with the *OEM*'s brand name. The *OEM* is able to establish an identity key with the TPMs that allows it to identify an original device and track them. Second, counterfeit devices must be distinguishable to the *OEM* and (depending on the type of device) remote detection of deployed counterfeit products can be necessary. The *OEM* is able to distinguish counterfeit devices from original ones even remotely. Note the potential for Relay Attacks on this. Third, *End User* need to be able to distinguish counterfeits from original products. This might require the help of the *OEM*. Similar to the above requirements, the *End User* can request a device validation by the *OEM*, e.g. via a device management platform. *End User* help via e.g. a management platform helps in the detection of Relay Attacks. Fourth, the *End User* should to be able to recognize counterfeits using an offline scheme without active contribution of the *OEM*. If the *OEM* provides certificates for the public portions of the TPM established identity keys, then the *End User* can validate the originality of a given device without active contribution of the *OEM*.

5.2 Additional TPM functionalities

Given the installation of a TPM there are additional functionalities that can be leveraged by the *OEM* as well as *End User*. As described in Section 3, one of the most valuable features of the TPM are the *RTM* and the *RTR*. These features allow the *OEM* and the *End User* to assert that a given version of the firmware or configuration is booted on a given device. These may be utilized for reporting but also in order to protect credentials that the device possesses against disclosure or misuse. Note that these depend on the integrity of the RTM as discussed in the section on Residual Attacks below. Furthermore, the TPM offers a certain amount of secure storage space (Non-Volatile memory). This memory can be used for arbitrary data and protected via several means, including passwords, software configuration and advanced policies in case of the TPM 2.0. Finally, there exist slight differences between the possibilities of TPM ownership between TPM 1.2 and TPM 2.0.

TPM 1.2 Ownership In the case of a TPM 1.2 deployment, the *OEM* is required to take ownership of the TPM in order to deploy its keys. Though this still allows *End User* to change the authentication values and use the TPM for their own purposes additionally, it means that a clearing of the ownership is not possible. This may pose a problem for resale of used devices, since a clearing of ownership is the only way to protect all TPM-bound resources from reuse by the future new device owner. However, the scheme for Remote Establishment of TPM1.2 as described above could be reenacted for a resale scenario. The *OEM* would need to employ an according functionality in its management software.

TPM 2.0 Ownership With the introduction of the Platform Hierarchy in TPM 2.0, it is now possible to completely separate the ownerships between *OEM* and *End User*. Accordingly, the *End User* may clear their ownership for resale and a new device owner make claim ownership of the TPM again without gaining access to the previous owner's secret material. Given the standard scheme for TPMs in consumer electronics, the *End User* and *OEM* interactions with the different platforms are distributed between OS and firmware. For devices as discussed in this scenario however these distributions may be realized differently, since there is no actual general purpose OS installation planned.

5.3 Residual Threats

Given the presented solutions, the following residual attacks still pose a (minor) threat. These threats are now discussed and their practical relevance evaluated.

Root of Trust for Measurement Manipulation The integrity of the RTM is under the direct responsibility of the *Device Assembler*. Since the *Device Assembler* acts as one of the potential attackers in this scenario, it seems contradictory to entrust it with this integrity requirement. However, the given scenario in which the *Device Assembler* is a potential attacker was solved without the use of the RTM. The RTM is intended for leveraging the additional TPM functionalities as described above. For these scenarios, a potential mistrust in the *Device Assembler* cannot be technically solved anyways, since the *Device Assembler* may always add additional components into a given device that provide so-called "backdoors". This can create a majour risk for devices in critical infrastructures and cannot be prevented by a TPM. However, the identification of the device based on the TPM will not be affected and, accordingly, the risk potential for successful attacks on the mechanism introduced in teh paper is not as high, but should of course be assessed.

Relay Attacks A relay attack could be performed from counterfeit products in such a way that devices are assembled without a TPM and any TPM-related request to this device if forwarded to an original device. In this case, however the counterfeiter would require one original device per counterfeited device, which would increase the attack costs beyond the counterfeiter's gain, and could be detected by the *End User* by the additional communication channel from the counterfeited device to the original device. If the counterfeiter would attempt to relay several counterfeited devices to the same original device (which would be profitable), then the *End User* could still detect the additional communication and the *OEM* would recognize the n-to-1 mapping between devices and TPMs, since e.g. a firmware update would be applied n times for only one device.

6 Conclusion and Outlook

TPM chips in the current version 1.2 as well as in the upcoming generation 2.0 provides a large set of hardware-based security functionality. It is a generic mass-market product and billions of devices are estimated to be currently equipped

with a TPM. Many of these TPMs are installed in PCs, where they are mostly not used. The counterfeit scenario concentrates on other types of devices, such as routers, embedded systems and other special purpose devices, where hardware-based security functionalities are are often not readily available. The TPM-based approach to counterfeit protection describes a new use-case for this security chip. The process shows that the security functionality provided by the TPM is suitable to establish counterfeit prevention and detection for outsourcing scenarios.

It should be noted that the integration of a TPM as well as the implementation of the initial key enrollment process as part of the anti-counterfeit scheme can be the basis for the efficient and cost-effective realisation of various additional security processes. One example is the remote attestation of software actually running on the system by feeding measurement information into the TPM through a Integrity Measurement Architecture (IMA) [7] and using TPM_Quote to generate securely report these measurements. This information can then be validated by comparing it to the expected status available through current remote software management systems that only know about software installed through the regular process. Changes due to attacks to the device will be recognized by changed hash values for the software that is loaded. into memory and reported by the TPM. Protocols like Trusted Network Connect or trusted mobile ad-hoc networks [8] use this feature to continuously or regularly determine the health by checking the software running with reference values.

Another example of an efficient management process based on the TPM results from the ability to automatically distribute initial configurations to the device without the need to touch it. This zero touch configuration protocol [6] only requires network access and can be implemented without any pre-configuration for particular clients. Also credentials for user authentication can be protected using the TPM. In the context of future production environments and cross-organization processes, a multi purpose trust anchor like the TPM can also be used to implement strong concepts for intellectual property protection.

References

1. J. Stradley and D. Karraker, "The electronic part supply chain and risks of counterfeit parts in defense applications," *Components and Packaging Technologies, IEEE Transactions on*, vol. 29, no. 3, pp. 703–705, 2006.
2. S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of puf-based "unclonable" rfid ics for anti-counterfeiting and security applications," in *RFID, 2008 IEEE International Conference on.* IEEE, 2008.
3. C. Mitchell, C. Mitchell, and C. Mitchell, *Trusted computing.* Springer, 2005.
4. B. Parno, J. M. McCune, and A. Perrig, "Bootstrapping trust in commodity computers," in *Security and Privacy (SP), 2010 IEEE Symposium on.* IEEE, 2010.
5. E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM conference on Computer and communications security.* ACM, 2004.
6. N. Kuntze and C. Rudolph, "On the automatic establishment of security relations for devices," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management.* IFIP/IEEE, 2013.

7. R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a tcg-based integrity measurement architecture," in *Proceedings of the 13th conference on USENIX Security Symposium*, vol. 13, 2004, pp. 16–16.
8. A. Oberle, A. Rein, N. Kuntze, R. Carsten, J. Paatero, L. Andrew, and P. Racz, "Integrating Trust Establishment into Routing Protocols of Today's MANETs," in *2013 IEEE Wireless Communications and Networking Conference (WCNC 2013)*, Shanghai, China, April 2013, pp. 1403–1408.