DELIVERABLE D4.1

Security requirements and specification for docking station module

Title of Contract	Seamless Communication for Crisis Management
Acronym	SECRICOM
Contract Number	FP7-SEC-218123
Start date of the project	1st September 2008
Duration	44 months, until 30th April 2012
Date of preparation	April 2009
Author(s)	Branislav Šimo, Zoltán Balogh, Daniel Hein, Peter Danner, Oscar López, Mikel Uriarte, Vladimír Hudek
Responsible of the deliverable	UI SAV, Branislav Šimo
Email	branislav.simo@savba.sk
Reviewed by:	Richmond Edwards, QinetiQ
Status of the Document:	Final
Version	1.3
Dissemination level	PU Public

Contents

1 Summary

This document provides a security requirements and specification based on the analysis of the results from WP2, where the functionality and technologies for distributed agent system are identified. It translates them into detailed functionality requirements for two hardware components – Secure Docking Module and Trusted Docking Station –reflecting the security mechanisms compliant with industrial standards related to distributed environments and security mechanisms performance. It further states requirements on the software environment and integration with specific components developed in other work packages.

2 Introduction

This document presents the security requirements and specification for the docking station module – secured and trusted software agent execution device – in the context of the secure agent infrastructure (SAI). The purpose of the docking station module as envisioned in the project proposal is to enable highly secure deployment and execution of software agents on remote machines, in order to permit information provisioning from different legacy information systems for the crisis management authorities and software. This is to make possible one of the main objectives of the SECRICOM project, to "add new smart functions to existing services which will make the communication more effective and helpful for users. Smart functions will be provided by distributed IT systems based on an agents' infrastructure." By fulfilling the above stated goal we create a pervasive and trusted communication infrastructure satisfying the requirements of crisis management authorities and ready for immediate application. Further, SAI should provide a secure distributed agent paradigm to achieve confidentiality and access to resources. It should also provide a smart negotiating system for parameterization and independent handling of access requests to achieve rapid reaction.

After the analysis of the internal and external requirements [D2.1] for the secure agent system and consideration of possible hardware and software solutions it has been decided that the functionality of what has been named as the "docking station module" will be realized by two complementary devices called the *Secure Docking Module* and the *Trusted Docking Stations*. Both of them will be further described and elaborated on in the rest of this document.

The *Secure Docking Module (SDM)* is a key storage device with local attestation and verification capabilities. The SDM protects a small set of key pairs for asymmetric cryptography. The SDM's key protection facilities are a standard function, which could already be implemented with today's smart cards or hardware security modules. The SDM extends this standard function by only releasing these keys to a host device if and only if this host device is in a *trusted state*. The host devices are called *Trusted Docking Stations* (TDS) and the process of establishing their state is called local attestation. The relationship between SDM and TDS is depicted in Figure 1.

Figure 1: The relationship between the Secure Docking Module and the Trusted Docking Station

A *trusted state* is a specific software configuration. This software configuration is measured by using a *Trusted Platform Module (TPM)*. A TPM is a special security chip which provides amongst other functionalities the protected capability of measuring the software

configuration of its host device. A TPM must be present in the TDS. The combination of a SDM and a TDS is called a *Secure Docking Station* (SDS).

The idea of the SDM/TDS concept is that if a TDS is in a *trusted state*, it can be trusted to adhere to a specific *policy*. A *policy* is a set of rules that constrains the behavior of a device for all conceivable situations. **This gives the TDS the freedom to execute any program that can be run as part of a *trusted state***.

2.1 *Example Use Case*

It is expressed several times in the Analysis of Crisis Management System Requirements deliverable [D2.2] that one of the most important things during the crisis management is timely delivery of relevant information in a secure and confidential way. The secure agent infrastructure is responsible for obtaining the information about available resources that are needed and helping the authorities manage the distribution of such resources (material or human).

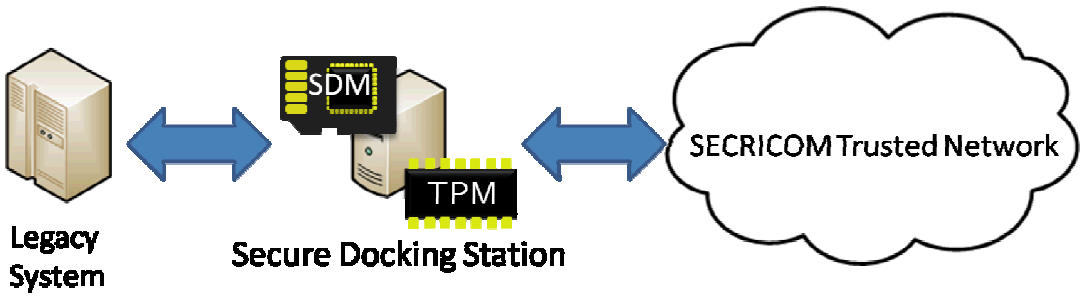


Figure 2: Legacy System application of Secure Docking Station

In order to do this it needs to communicate with various legacy information systems operated by agencies and institutions involved in the crisis resolution. Such legacy information systems (LIS) often contain sensitive information and must be secured from misuse and exposure to the outside world. SECRICOM will address this by deploying a Secure Docking Station (SDS) – a combination of TDS and SDM – into a physical proximity of the legacy information system, preferably in the same room. The SDS then acts as a secured and trusted extension of the SECRICOM infrastructure. This configuration eliminates the exposure of the LIS to the outside world and allows the operator of the LIS to have increased trust in the information consuming party. The SDS, being an agent execution platform, can even perform processing of the information received from the LIS and thus conserving the bandwidth of the network and the possible exposure of the sensitive data, sending back only the results. The remote deployment of the SDS also helps to overcome the network connection problems, because the data processing on the SDS can continue even if there is a temporary loss of connection to the outside world.

3 Secure Agent Infrastructure Requirements

This paragraph describes the security-related issues and security requirements of the Secure Agent Infrastructure (SAI).

The role of agents in the SECRICOM project is primarily coordinated collection of information. The gathering of information is enacted either from legacy systems or from human end-users through mobile devices by guided dialog. In respect to requirements the overall agent infrastructure must be a secure, robust and fail resistant system. Agent technology was selected due to the ability to fulfill such requirements through support of mobile and dynamically deployable executable code.

3.1 Infrastructure Security Requirements

In order to define concrete security requirements we must sketch the basic infrastructure in which agents will operate (Figure 3). The network of Trusted Servers (TS) is the home platform for agents. According to [Jans01] the platform from which an agent originates is referred to as the ***home platform***, and normally is the most trusted environment for an agent. This is also true for SECRICOM agents – the network of TS is a managed set of systems with defined security policies and possibly managed by a central authority. From here agents are delegated to host platforms to gather data and information. TS host core services of the agent platform. Agents are mainly executed on remote sites which provide the computational environment in which agents operate. We will refer to these sites as to ***host platforms*** *(or agent platforms)*.

In general, any party which wishes to join the SECRICOM system and to provide information from their legacy systems or users must introduce a host platform for agents. We will refer to such parties as Host Platform Providers (HPP). From end-user requirements, the following HPPs were identified so far (Figure 3):

- *Resource Providers* – hospitals, fire brigade, police, warehouses or any other entities which can play a role in the mitigation of crisis situation,
- *Command Centers* – mobile (nomadic) centers which coordinate locally the incident site;
- *General Command Center* and *Operators* – usually located in one place or at least tightly interconnected.

The features of SECRICOM agents will encompass several carefully chosen attributes:

- *Code mobility* (without execution state) – ability to move code to different platforms and execute there, within the project we do not plan to support execution state mobility (as there is no requirement for that),
- *Autonomy* – ability to deliver gathered data to one or several optional destinations,

- *Reactivity* – in some cases agents will perceive the context in which they operate and react to it appropriately (e.g. agents can monitor availability of some resource and notify the requestor).

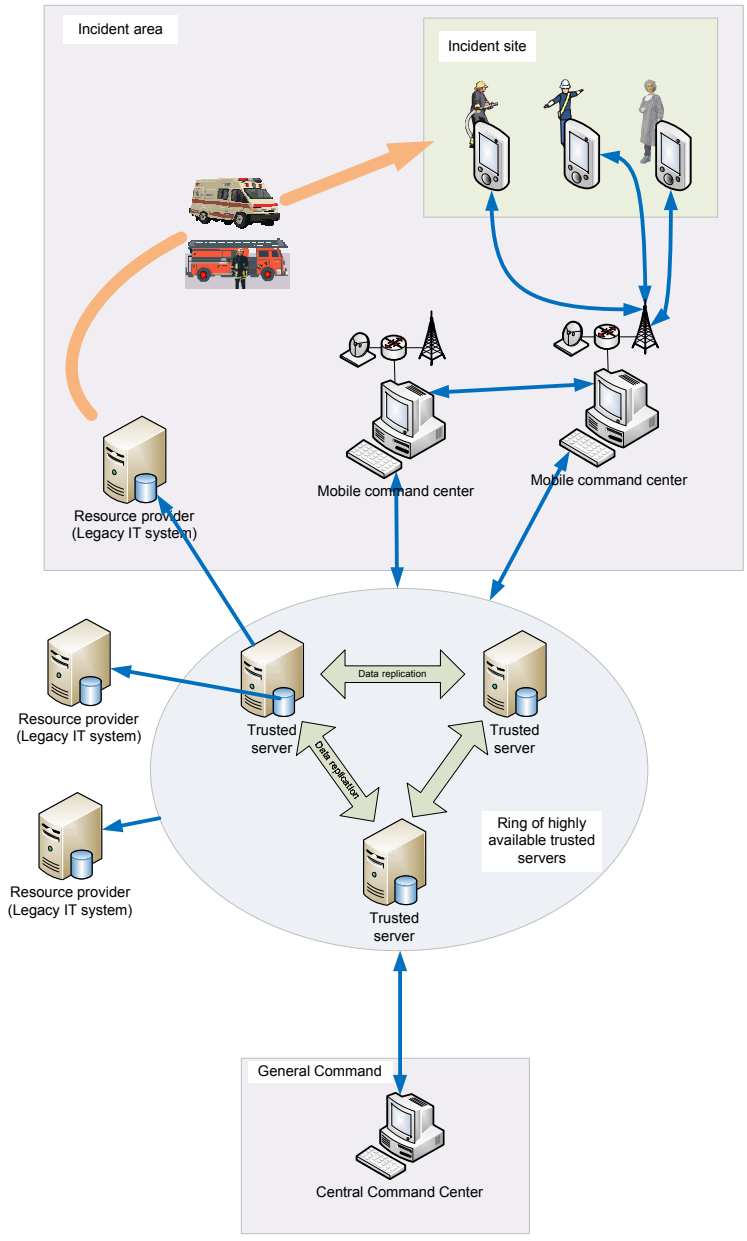


Figure 3: Host Platform Providers for the Secure Agent Infrastructure.

Since agents collect information which is often of high sensitivity, confidentiality and security, while at the same time requirements for action or decision traceability exist, agents must be provided with a secure, trusted and attested execution environment. In the following we identify main agent-related security threats. A detailed explanation of generic mobile agent security aspects is discussed in [Jans01]. Generally, four threat categories are identified:

- Agent platform attacking an agent,
- Agent attacking an agent platform,
- Agent attacking another agent on the agent platform,

- Other entities attacking the agent system.

The last category covers the cases of an agent attacking an agent on another agent platform, and of an agent platform attacking another platform, since these attacks are primarily focused on the communications capability of the platform to exploit potential vulnerabilities. The last category also includes more conventional attacks against the underlying operating system of the agent platform.

3.1.1 The Host Platform Attacking the Agent

The main threat for agents in foreign execution environment of host platforms is the "malicious host problem". This is one of the main problems in the class of "an agent platform attacking an agent". Simple explanation of "malicious host problem" is provided in [Bor01]: "Once an agent has arrived at a host, little can be done to stop the host from treating the agent as it likes". Therefore, the main requirements from the agent-side are laid out in respect to the "malicious host problem". Concrete security requirements of agents in respect to the host platform are as follows:

- Isolated execution environment for agent execution – not only virtual isolated execution environment but dedicated isolated hardware preferred;
- Means to attest the platform required in order to detect if the host platform is in trusted state;
- Protected storage for credential data (such as PKI's secret key).

3.1.2 The Agent Attacking the Host Platform

There are also threats stemming from an agent attacking an agent host platform. Therefore reversely a host platform has also requirements in respect to agents. These requirements are more evident when provided in context of HPPs security requirements:

1. HPPs do not want to install and execute any external application (including SECRICOM system) on their systems in line with their strategic legacy applications.
2. HPPs prefer to have a dedicated and isolated system for SECRICOM which would connect to their legacy system in a secure predefined way.
3. HPPs want to be able to control what (data), when and by who (traceability) is provided to the SECRICOM system.
4. HPPs want to be able to configure set of applications executable on their side. Agents must be therefore audited and verified thus mediate trust to executable agent code.

The agent platform has the following security requirements in respect to agents:

- Isolated execution environment for agent execution - agents must be executed in isolated environment (isolated hardware preferred), so an agent can not harm legacy systems;
- Means to monitor and trace agents activity;

- Means to configure the set of agents executable on the host platform;

In order to track agents, any agent in the platform must be cryptographically signed. Only agents signed with trusted authority and assigned to selected category will be trusted by a host system.

Agents need to send signed messages to Trusted Servers.

3.1.3 The Agent Attacking another Agent

It is required that any agent which will be used in SECRICOM will need to be audited and certified by a central authority. In turn every host platform will be configured to execute only agents which are certified. These two security policies should ensure that malicious agents will not be deployed into the infrastructure. Only a breach of the set security policies might lead to potential agent-to-agent security risk.

Moreover, each agent will be executed in a relatively isolated virtual environment with limited access to data of other parallel executed agents on the same host platform.

3.1.4 Other Entities Attacking the Agent System

Agents will also connect to legacy systems (third party software). Therefore a risk of an agent being attacked by a legacy system but also vice versa – the risk of attacking legacy system by an agent also exists.

The host platforms will need to provide some kind of connection to legacy systems. We explicitly presume that this will be a network connection. On any network connection there is an eavesdropping risk. Therefore another requirement which arises from agents to the host platform is:

- Secure protected connection to legacy systems.

Physical security of network connection can be achieved either by direct cable connection of the host platform with legacy system or by managed network security (managed switch with well defined security policies). The data transport security will be achieved primarily through encryption.

3.2 Agent Life Cycle and Related Security Requirements

The life cycle of an agent in the secure agent infrastructure (SAI) is the primary source of security requirements of the SAI. The creation of an agent encompasses development of its code, audit and certification. After successful certification of its code it is equipped with a private key, which is (for all of its existence) available only to the agent itself. A corresponding public key is stored and accessible through the SECRICOM PKI. After its certification and "priming" with a private key, it is stored in an agent registry (AR). From this registry, it is downloaded to a TDS which needs to use the agent's capabilities. The downloaded copy has to be secure at all times – during transfer from the AR to the requesting TDS, and also during its deployment and execution inside the TDS. The copy inside TDS is destroyed when it finishes executing and delivers its results. The life cycle of an agent ends when its certificate expires, and after this it may be deleted from the AR, since

it cannot be deployed anywhere in the SAI anymore. See Figure 4 for a graphical representation of this process.

The life cycle of a SECRICOM agent sets the following requirements on the security infrastructure of SECRICOM:

- A SECRICOM agent must contain its private key; this key must not be known to any other entity during the whole lifetime of the agent.
- A SECRICOM agent must be audited before it can be used; the audit must ensure, that the agent does only what its creator states it should do, and that it does not contain any malicious code, which may jeopardize the integrity of the SECRICOM environment.
- A SECRICOM agent must be protected at all times from revealing its private key; it must always be either stored in a trusted device, or encrypted when it is outside of such device.
- Each audited agent must be issued a certificate, signed by its auditor, which states the capabilities of the agent as specified by its creator and verified by the auditor.
- The SECRICOM infrastructure must contain a service for storing and accessing certificates of entities inside the infrastructure; one of the classes of these entities is the class of the SECRICOM software agents.
- All results produced by a software agent must be protected from being revealed to any entity different than their intended recipient – the client. Also, it must be asserted that their authenticity can be verified by the client upon their reception.
- The results provided by an agent must be signed by both the agent and the device running the agent's code in order to ensure the trust in the results by the Process Management System. Each secured device must be connected to SDM providing the encryption keys authenticating the device and its user, respectively.

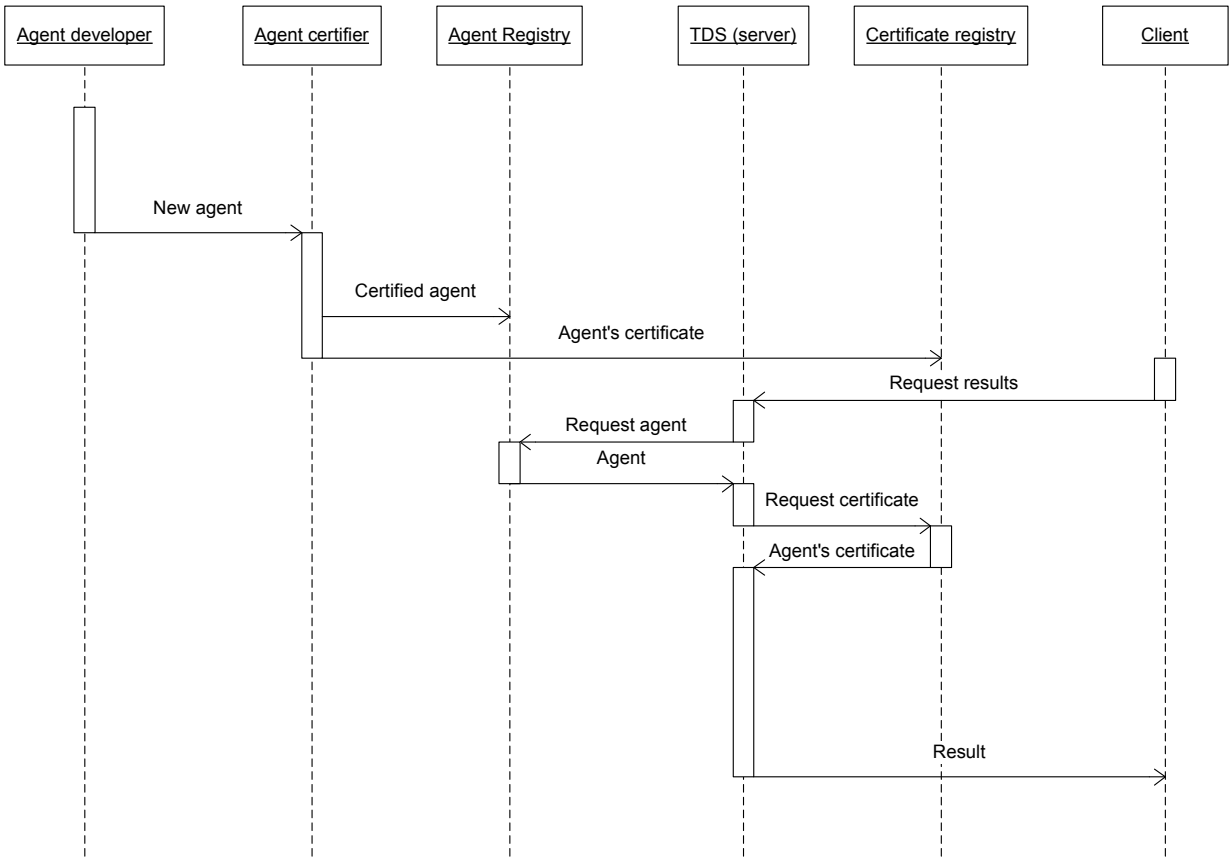


Figure 4: Agent life cycle

3.3 Agent Registry

Agent Registry (AR) is a service which stores all the existing software agents in the SECRICOM infrastructure. The registry itself is implemented by an agent, which resides inside a TDS. The registry must ensure that any agent stored inside it is secure, and will be handled in a manner which will not reveal the secrets it contains to none but the authorized parties. AR has the following requirements on the security infrastructure of SECRICOM:

- Any request for a software agent to be downloaded from the registry and deployed inside a device must clearly state the recipient of the agent.
- All devices in the SECRICOM infrastructure (TDS and others) must be issued a certificate stating which agents it may receive; AR will reveal to a device only such agents, and will deny the deployment of agents for which the device is not certified.
- The agent intended for deployment in a device must be protected during transport from being revealed to third parties; it must be encrypted in a manner which allows only the specific pair of a device and an agent, to which it is addressed, to be able to decrypt it and execute it.

3.4 *Access to Legacy Systems*

A *legacy system* is an already existing information infrastructure related to crisis management. Agencies operating legacy systems that contain confidential data such as health records need to ensure that their data is handled according to the agreed policies [D2.1]. Such assurance is based on the trust in the SECRICOM system operator and the trust in the security of the system he is using. We can influence only the latter.

3.4.1 Security Considerations

Envision is an e-Health application that grants access to medical records and allows requests for ambulances, number of beds available in hospitals, etc. Such an application requires a mutual trust relationship between the TDS and the legacy system. The legacy system has to trust the DSAP to enforce data security and privacy policies and that all requests for services are genuine. On the other hand the crisis management has to be assured that the received data is genuine and that requests for services are processed in a correct manner. One of the tasks of the SECRICOM agents is to limit the exposure of sensitive data to third-party entities by collecting and processing the data from the legacy system inside of the TDS connected to the system and passing forward only the results of the processing.

The Distributed Secure Agent Platform (DSAP) together with its Resource Inquire System (RIS) subsystem provides interfaces and protocols for communication with legacy information systems (LIS). The actual communication with a LIS will be performed by the Information Delivery Agents (IDA). RIS will provide pointers (or addresses) to secure systems which can be queried by an IDA in order to retrieve information about certain resources. Legacy systems are contacted via an Information Delivery Agent (IDA) in order to obtain information about resource availability and/or request resource deployment. The examples of information types needed for generic and specific crisis management planning are: transportation, emergency personnel (medical, firefighters and police), equipment, tools, etc. Resource queries can range from simple inquiries regarding available numbers of specific resources to complex queries over (relational) databases (e.g. finding out the number of people affected from the database of residents, identifying possible immobile or otherwise handicapped people by consulting the medical records database, etc.) or queries to document stores (e.g. search and delivery of pertinent instructions to crisis teams, mapping information, etc.).

The IDAs will run on the Trusted Docking Station (TDS). It is therefore important to ensure that the TDS is in a trusted state and that it enforces the policies required by the legacy system operator and/or by the law. To further increase the trust, the TDS should be placed physically close to the legacy system. The protection of cryptographic material performed by the SDM attached to the TDS might include credentials or keys that allow access to the services of the legacy system. However, the exact conditions of the access to the legacy system are subject to the capabilities of the legacy system and the mutual agreement between the operators of the legacy system and the SECRICOM system.

3.4.2 Healthcare Example

One example of a legacy system type that SECRICOM shall deal with is a Hospital Information System (HIS). As they handle and store sensitive medical and personal information, there are specific requirements on security management. There are three basic areas (issues) in security of HIS [Fer07]:

- *Confidentiality* – the prevention of unauthorized disclosure of the information
- *Integrity* – the prevention of unauthorized modification of the information
- *Availability* – the prevention of unauthorized withholding of the information.

Confidentiality is often used interchangeably with privacy, but they are not exactly the same. Privacy is the right of an individual to not have their private information exposed, whilst confidentiality is limiting access to information to authorized individuals only.

HIS Information Security Policy usually has to deal with the following areas:

- Establishment of HIS wide approach to information security management system.
- Prescription of mechanisms that help identify and prevent the compromise of information security and the misuse of data, applications, networks, computers, printers scanners and all other asset owned by HIS.
- Definition of mechanisms that protect the HIS assets till satisfy its legal and ethical responsibilities with regard to its networks and computer systems connectivity to internal and external networks.
- Prescription of an effective mechanism for responding to external complaints and queries about real or perceived non-compliance with this policy.
- Ensure that Information Resource Controls are in place, are effective, and are not being bypassed.
- Ensure the achievement of Audit Compliance, Service Level Monitoring, Performance Measuring, Limiting Liability, and Capacity Planning.

Paper [Fer07] gives an overview on the access control and security management applied specifically in healthcare environment. There are also initiatives to standardize and formalize HIS development. CEN EN 13 606 (Electronic Health Record Communication) – European standard that defines interoperability architecture for Electronic Medical Systems, HL7 [1] - provides standards for interoperability that improve care delivery, optimize workflow, reduce ambiguity and enhance knowledge transfer. These standards should be considered while developing connections to HIS.

[1] http://www.hl7.org/

3.5 *Infrastructure Monitoring*

Infrastructure monitoring is an important factor in keeping the infrastructure security and availability. The monitoring information will be gathered and processed by the Communication Monitoring and Control Centre (CMCC) – to be developed in WP9.

The concept of TDS as a computing device that will execute agents and the requirement that TDS reach a *trusted state* will be one of the aspects that have to be measured.

3.5.1 Monitoring from the point of view of agent and state of TDS

The CMCC will be used as an Agent Monitoring Data Collector, so some defined parameters will be recollected from the agent infrastructure deployed and running on TDS.

From this perspective and in terms of understanding the role of agents as a distributed collection of information the CMCC will gather:

- Number of agents executing on TDS
- IDs of the agents
- Type of agents
- Type of agents deployed in the infrastructure (Fixed, Nomadic, Mobile)
- Log and accounting of agents activity
- State of the AEE (Agent Execution Environment) task completed or not completed
- TDS state (trusted or not trusted)
- Number of TDS on trusted state

3.5.2 Concrete information which could be monitored

It will be necessary to determine the set of the possible functionalities and possible priorities of implementation:

- Date and time of agent delegation and agent task completion, or other state parameters
- Status of an agent (active, error connecting to legacy system, connection lost, etc.). One thing which CMCC could allow here is to monitor the bandwidth available to individual agents and if the agent reports a "bandwidth not sufficient" status, the CMCC operator should allow to the bandwidth on a specific MBR to be raised (agent – operator bandwidth negotiation).
- Expected and real time agent task completion;
- Speed of response (QoS in terms of response delivery);
- Monitoring of data flows (from/to) agents;
- Number of agents deployed from individual TDS and/or to Fixed, Nomadic or Mobile

A list of other information to be monitored, which is not directly related to agents is provided in Appendix A – Other TDS monitoring parameters.

4 Requirements for Secure Docking Module

The Secure Docking Module (SDM) must provide two interlocked capabilities. On the one hand it is a secure key storage, while on the other hand it is a local attestation device. By establishing the trusted state of the host device it is possible to leverage the computing power of the host. Therefore the SDM does not provide an on chip computing environment.

The SDM establishes the trusted state of the Trusted Docking Station (TDS). According to the SECRICOM DoW the SDM is a single chip add-on module. The process of establishing the trusted state of a platform with an add-on module is a procedure that involves multiple parties! It cannot be achieved by an add-on module alone! Therefore, it is necessary to modify the host platform. In order to minimize the necessary modifications of the host device the existing TPM technology must be used.

SDM protected keys must only be released, when the TDS is in a trusted state. To ensure this a protocol must be defined that governs this key release process. A preliminary version of the protocol is shown in Figure 5.

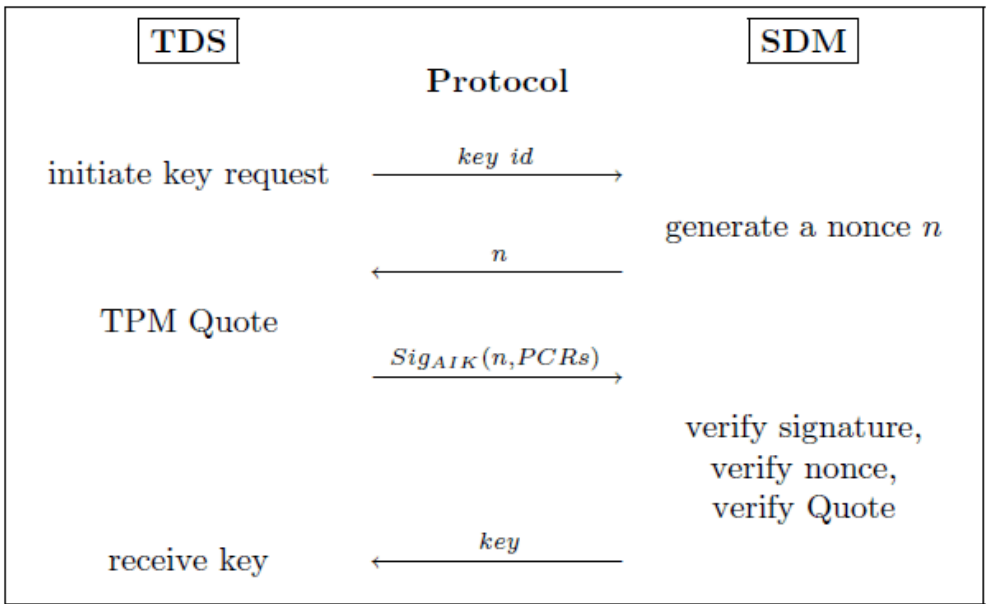


Figure 5: The core SDM key release protocol, preliminary version.

The basic key release process works as follows. A software program on the TDS requires a key. This key could for example protect the agents in the agent repository and the requesting software could be the agent repository agent. In order to retrieve the key from the SDM it issues a key request to the SDM. This key request must at least contain the ID of the key. The SDM generates a nonce and sends it back to the TDS. The TDS uses its TPM to measure its configuration and create a signed platform configuration report. This platform configuration report is called a *Quote*. The *Quote* incorporates the nonce from the SDM to guarantee the freshness of the report. The report is signed with a special premeditated key, the so called Attestation Identity Key (AIK). This key is a 2048 bit RSA key managed

and protected by the TPM in the TDS. The SDM receives the signed *Quote* and verifies the signature, the nonce and the *Quote* itself. If, and only if, all three checks succeed the requested key is released to the TDS.

The key release protocol as described above is a preliminary concept and should serve as a base for the functionality of the SDM. The exact functional specification of the SDM is part of WP5 **D5.1 Functional Specification.**

The protocol outlined in the Figure 5 version is vulnerable to eavesdropping attacks. It is possible to eavesdrop on the communication between TDS and SDM and thus obtain the key. Therefore, the key release protocol must be protected by an end-to-end secure communications channel. The security of this channel must be based on standard cryptographic primitives such as RSA public key cryptography, AES symmetric cryptography and standardized hash functions.

The use of a TPM imposes two security limitations. First, the TPM signs the *Quote* with a 2048 bit RSA key. Therefore, the security of the protocol is as secure as the digital signature created by this key. Any key protected by this local attestation procedure is therefore limited by this security level. Consider for example an SDM that protects a 4096 bit RSA key in its protected storage. In order to obtain this key it is necessary to break a 2048 bit RSA key. Therefore, under certain circumstances, the 4096 bit RSA key does not provide additional security compared to the 2048 bit key. The aforementioned circumstances occur if the attacker can attack the SDM/TDS directly.

The second limitation imposed by the TPM is the level of hardware attack protection required of a TPM. According to [Gra06] a TPM should withstand simple, limited physical attacks. The security of the key release protocol is directly linked to the security of the TPM. Therefore, the SDM must be at least as secure against hardware attacks as the TPM. Thus it should withstand simple hardware attacks.

In conclusion the security requirements for the SDM are:

- A tamper resilient memory region to store the SDM protected keys. The required level of resilience is limited by the resilience of the TPM in the TDS.
- An authenticated and encrypted communications channel between the SDM and the client software that requires SDM protected key material. The security of the key release protocol relies on a digital signature with a 2048 bit RSA key. The same security level should be used to protect the communications channel.

4.1 HW Specification

As a result of the recent efforts in WP5 it is possible to outline a HW specification. The HW specification given here is preliminary. The exact hardware configuration of the SDM will be specified as part of WP5 task **T5.2 (M9-M18): Design of the Secure Docking Module** (IFX, TUG, CEA, HIT). The list of required components that have been identified so far is as follows:

1. A processor and associated memory

A small processor with a moderate amount of RAM is required to control the SDM subcomponents. This combination may also implement some or all of the SDM functionality in software.

2. An RSA module

   The RSA module serves two purposes. Its primary purpose is to verify the signatures on the TPM *Quote* operations. Its secondary function is to help establishing an authenticated and encrypted communications channel between the SDM and its TDS host. To fulfill these requirements the RSA module must be capable of encryption, decryption, and creation and verification of digital signatures.

3. A SHA-1 hash module

   A TPM *Quote* is a digitally signed platform configuration report. The actual platform configuration is represented by a set of SHA-1 hashes. These SHA-1 hashes are stored in so called Platform Configuration Registers (PCRs) inside the TPM. To minimize the size of the *Quote* a SHA-1 hash of a selected set of PCRs is computed and subsequently signed by the TPM. Depending on the exact details of the key release protocol it might be necessary to recalculate the SHA-1 hash of the selected set of PCRs.

4. A True Random Number Generator

   The True Random Number Generator is required to generate numbers that are only used once, so called *nonces*. The key release protocol outlined in Figure 1Figure 5 for example requires an SDM created nonce to guarantee the freshness of the Quote generated by the TPM in the TDS.

5. A symmetric cryptographic primitive

   The necessity of an authenticated and encrypted communications channel requires a symmetric cryptographic primitive to encrypt the communication. An example of such a primitive is the Advanced Encryption Standard (AES) block cipher. The wide acceptance of AES as a cryptographic standard and the fact that there are no known exploitable weaknesses makes it AES a valid choice for this purpose.

6. A non-volatile storage for keys and platform configurations

   The primary purpose of the SDM is to protect key material for asymmetric cryptography. This key material is stored in a non-volatile memory on the SDM. The SDM must also store a set of valid platform configurations (*trusted states*).

7. A physical interface

   The physical interface defines how the SDM is physically connected to the TDS. Several considerations factor into the choice of a physical interface for the Secure Docking Module. These considerations are security, compatibility and bandwidth.

Thanks to the end-to-end secure channel security requirement the relevance of the physical interface to the security of the SDM is limited.

The SDM's primary function is to protect key material. This key material must be released whenever client software requires it. Therefore, the bandwidth requirement depends on the amount of transferred data per key release and the number of key releases per power cycle. The amount of transferred data is estimated to be about a few kilo bytes depending on the size of the key material. Once the SDM releases a key, the client software has full control over it. Therefore, it can be expected that client software initiates only one key release sequence per key and only once per life cycle. For these reasons, bandwidth is no limiting factor in the selection of the physical interface of the SDM.

The last point to consider for the interface is compatibility. The SDM should also provide limited services to mobile host devices and therefore a MicroSD interface variant of the SDM will be considered as part of the design process. For non mobile host devices, the current aim is to provide a USB interface.

4.2 *Functional Specification and Interfaces*

The primary function of the SDM is to protect cryptographic key material and to release it only if the host of the SDM, the *Trusted Docking Station*, is in a *trusted state*. The primary functionalities the SDM provides to a *Trusted Docking Station* can be grouped into two categories:

1. Releasing cryptographic key material to *Trusted Docking Stations* in *trusted states*.
2. Administrating the SDM key release functionality
    a. Adding/Removing *Trusted Docking Stations*
    b. Adding/Removing cryptographic key material for a specific *Trusted Docking Station*
    c. Adding/Removing trusted states to a specific cryptographic key material entry

The key release functionality is conceptually simple. The process is depicted in Figure 5 and has already been explained above. The administrative functions are actually more complex than the core functionality of the SDM itself. All specifications in this section are preliminary. The exact specifications will be part of deliverable **D5.1 Functional Specification of the Secure Docking Module**.

4.2.1 Specifications for communication

1. Authentication protocol

The first step to obtain a key from the SDM is to establish an authenticated, encrypted session with the SDM. Based on the deliberations in the HW Specification section, it is manifest to use the RSA module for authentication based on asymmetric cryptography. To

establish the session random values are exchanged between the SDM and the client software in the TDS. The process is depicted in Figure 6.

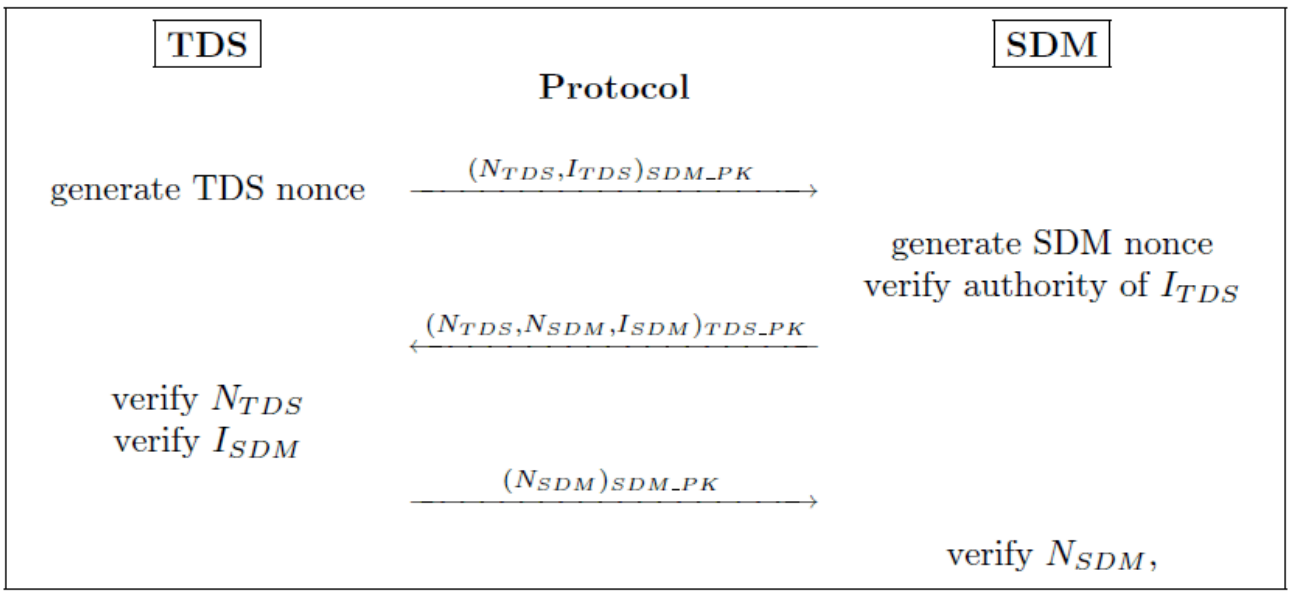


Figure 6: Session authentication protocol

The terms N_{TDS} and N_{SDM} denote the nonces generated by the *Trusted Docking Station* and the SDM respectively. The terms I_{TDS} and I_{SDM} describe the unique, certified identifiers of the *Trusted Docking Station* and the SDM. They are required for the SDM to select the correct encryption key. The messages in the protocol are encrypted using the public key of the recipient: *SDM_PK* and *TDS_PK*. In SDM terminology the public key of the SDM (*SDM_SK* in the figure) is called *SDM Authentication Key (SAK)*. The public key of a *Trusted Docking Station* (*TDS_PK SK* in the figure) is termed *Host Authentication Key (HAK)*.

2. Session encryption

The two nonces exchanged during the authentication protocol allow the generation of a session key. For this the two nonces are appended to each other and than a SHA-1 hash of the result is computed:

$$AES_{Session\ Key} = SHA\text{-}1(N_{TDS} \,||\, N_{SDM})$$

The thus determined session key will be used in conjunction with an *Advanced Encryption Standard (AES)* module to encrypt all traffic between the TDS and the SDM. For now we have chosen Cipher Block Chaining (CBC) mode of operation for the AES module. This requires the use of an Initialization Vector (IV). In the current concept this IV is provided by the SDM and send encrypted to the TDS (using the HAK of the *Trusted Docking Station*). For details on the cryptographic principles see [MvOV01].

4.2.2 Specifications for SDM data storage and maintenance

The SDM manages a set of data structures which contain the information required to provide the SDM capabilities. This primarily refers to key protection and platform attestation verification. This data needs to be structured in order to allow operations on it. The overall structure of the involved data structures is depicted in Figure 7.

The SDM provides a limited amount of protected storage. Therefore, the size of the actual data must be kept at a minimum. The key material data container and the information that encodes valid platform configurations under which keys are released are all of a fixed size. Many of the data structures stored on the SDM require unique identifiers to facilitate retrieval of information. These identifiers should be short, but still enable unique identification within the SECRICOM infrastructure.

3. Trusted Docking Station

The *Trusted Docking Station* data structure is illustrated as a light blue block in Figure 7. It encompasses a unique identifier, *Host Authentication Key (HAK)*, and an *Attestation Identity Key (AIK)*. The unique identifier is just for identification, it is not a certificate. The AIK is used by TPM of the *Trusted Docking Station* to sign a platform *Quote*. The five operations related to the *Trusted Docking Station* data structure are:

3.1. Add a new *Trusted Docking Station*. Creates a new, or updates an old *Trusted Docking Station* with the specified identifier, HAK and AIK.

3.2. Get a *Trusted Docking Station* by its identifier.

3.3. Remove a *Trusted Docking Station* selected by the specified identifier.

3.4. List all *Trusted Docking Station* stored in the SDM.

3.5. List all *Protected Key Materials* of a specific *Trusted Docking Station*. Provides a list of unique identifiers of all *Protected Key Materials*.

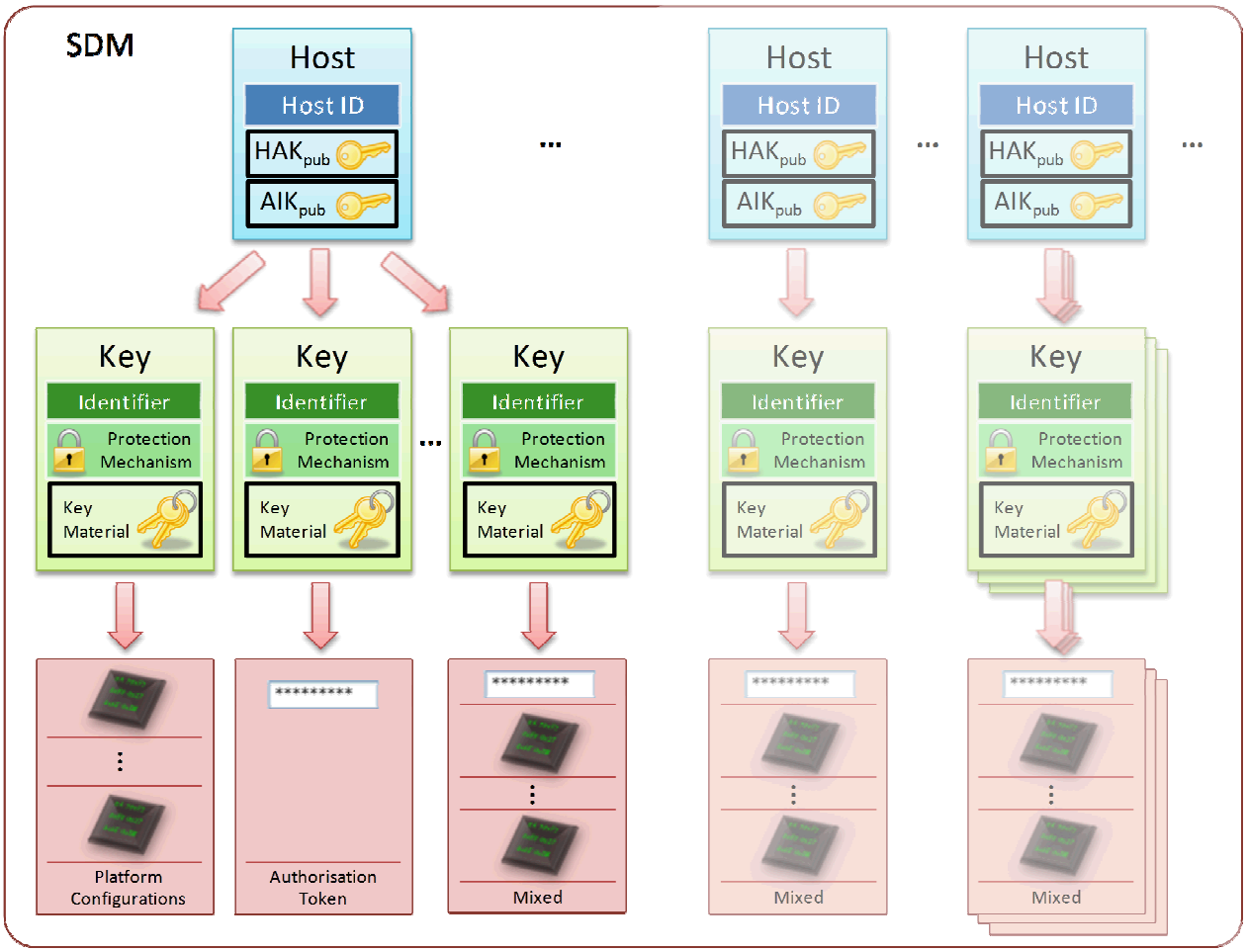


Figure 7: The relationship between Trusted Docking Stations, protected key material, and trusted states.

4. Protected Key Material

The *Protected Key Material* data structure is comprised of a protection type, a unique identifier, the actual key material, and the associated protection mechanism information. There are two key protection mechanisms. The first is the release of keys only to platforms which are in a specific state. In order for the SDM to support devices that are not equipped with a TPM, the SDM supports a second key protection mechanism. This mechanism requires some kind of authorization token, which is simply a password or PIN, to release a specific key. This capability is specifically for mobile devices without a TPM, but with a user present.

The unique identifier of a key material structure allows lookup of the *Protected Key Material*. The actual key material is a byte array, currently with a maximal size of 1024 bytes. The associated protection mechanism is either a list of valid platform configurations (Platform Configuration type), or an authorization token (Authorization Token type). Therefore, depending on the type of protection mechanism, the SDM provides a different API:

4.1. Add a new *Protected Key Material*. Creates a new, or updates an old Protected Key Material using the specified unique identifier, protection mechanism type and concrete key material.

4.2. Get an existing *Protected Key Material* identified by the specified unique identifier.

4.3. Remove an existing *Protected Key Material* identified by the specified unique identifier.

4.4. In case the *Protected Key Material* is of type Authorization Token the add/update function must also specify an authorization token.

4.5. If the *Protected Key Material* is of type Platform Configuration functions 4.1 to 4.3 are not modified, but there must exist a function to list all *Valid Platform Configurations* of a specific *Protected Key Material*.

5. Valid Platform Configuration

A Valid Platform Configuration is a 20 byte SHA-1 hash value. A *Protected Key Material* of type Platform Configuration may specify a set of valid platform configurations. The API must specify functions to add, remove, and test the existence of *Valid Platform Configurations*:

5.1. Add a new *Valid Platform Configuration*. Notice, that it does not make sense to update an existing platform configuration.

5.2. Remove an existing Valid Platform Configuration specified by its 20 byte value.

5.3. Test, if a *Valid Platform Configuration* with a specific 20 byte value exists. Providing a function to get the configuration does not make sense, because a *Valid Platform Configuration* consist only of its 20 byte value, which is also a kind of unique identifier for the valid configuration.

Data Type	Size [Bytes]
Unique Identifier	20
RSA Public Key	512
RSA Private Key	512
Key Material	1024
Valid Platform Configuration	20
Authorization Token	20

Table 1: The list of data types known to the SDM.

Table 1 provides a list of all currently specified data types known to the SDM and the respective size. Unique Identifiers are used by *Trusted Docking Stations* and *Protected Key Materials*. RSA public keys are required by the *Trusted Docking Stations* to store the HAK and AIK of a *Trusted Docking Station*. Currently there is only one use for an RSA private key type, which is the private part of the SAK. *Key Material* is used by the *Protected Key Materials* to represent actual SDM protected key material, and *Valid Platform*

Configurations protect certain *Protected Key Materials*. *Authorization Tokens* are used to protect key material on systems that have no TPM, but have a user present.

4.3 *Emulator requirements*

To allow rapid development of the software components depending on the SDM a software emulator of the SDM is necessary. Furthermore, the SDM hardware development flow requires an exact functional specification. The well defined interface of a software emulator of the SDM can serve as such a functional specification. A third and important point is that for the SDM to achieve its full potential it is necessary to develop a set of middleware components and a library on the TDS side. The development of a prototype of the middleware and support libraries is also part of the emulator.

The SDM cannot operate without a middleware support component on the host system, the TDS. The structure and components of the middleware are illustrated by Figure 8: Components of the Secure Docking Station software architecture. To leverage the full potential of the SDM it is necessary for the host platform to enable the cooperation between the TPM and the SDM. The *Trust Management Component (TMC)* of the middleware provides this functionality.

The TMC provides a simple API for key retrieval. To actually retrieve a key it is necessary to determine the platform configuration, employ the TPM to create a signed report of the platform configuration, send the report to the SDM for validation and receive the key from the SDM.

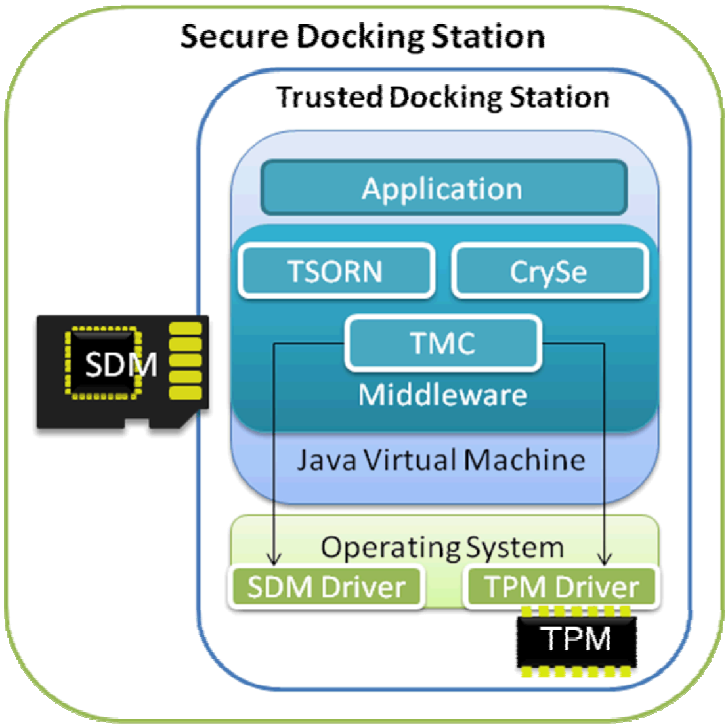


Figure 8: Components of the Secure Docking Station software architecture

5 Requirements for Trusted Docking Station

5.1 Introduction

A Trusted Docking Station (TDS) is a Commercial Off-The-Shelf (COTS) computer platform with specific hardware and software requirements. The hardware requirements are specified in section 5.2.2.

The primary requirement of a Trusted Docking Station is to execute software in a *trusted state*. The establishment of a *trusted state* has two prerequisites. The first is the capability to measure the exact software configuration of the platform. The second requirement is to shield the software executing in a *trusted state* from other software on the same machine. These two requirements are interwoven and necessitate a holistic solution approach.

5.1.1 Measurement

A TPM provides the basic capability to measure a platform. The force behind Trusted Computing and the TPM – the Trusted Computing Group (TCG) – has envisioned the measurement of the platform configuration by building a chain of trust[2]:

- At power on of a platform the so called Core Root of Trust for Measurement (CRTM) measures the BIOS of the Platform.
    - The measurement is a SHA-1 hash of the BIOS image.
    - The SHA-1 hash is stored into a protected location.
    - The TPM provides this protected location and it is called Platform Configuration Register (PCR).
- After having performed the basic initialization of the platform, the BIOS measures the boot-loader of the operating system, before transferring control to it.
    - The measurement of the boot loader is again a SHA-1 hash of the boot loader software image.
    - The SHA-1 hash is again stored into the TPM
- This process continues until a chain of trust is built from the first step in the boot process up to the running applications.

The important idea behind the chain of trust concept is to measure a component before transferring control to a component. Thus, it is discernible by analyzing the measurement of the next component if this component will continue this chain of measurement. A problem with the chain of trust concept is that the shielded location in the TPM that stores the platform configuration measurements (SHA-1 hashes) only provides very limited storage. This necessitates that PCRs are reused. It is not possible to simply overwrite a PCR value as this would eliminate the previous software measurement. To solve this problem

[2] The chain of trust concept has been simplified to abstract technical details that are not important for grasping the concept.

and to provide an effective measure against counterfeiting measurements the only way to change a PCR is concatenate the hash in the PCR with the new value and hash the result. This is enforced by the TPM.
This introduces the problem that the order in which measurements are taken becomes significant. Virtualization can mitigate this problem, which leads to the second requirement of achieving a *trusted state* protection against other software.

5.1.2 Virtualization

Virtualization provides an abstraction of a physical platform that is known as a *Virtual Machine*. The term virtualization encompasses a variety of virtualization techniques. For the purpose of this document we use the term to refer to a fully virtualized platform. A fully virtualized platform provides two important capabilities:

- It abstracts the physical characteristics of physical platform
- It provides isolations of the *Virtual Machines*

The term *Virtual Machine* is also used in conjunction with the Java programming language and the Java Runtime Environment. To avoid confusion a *Virtual Machine* in the context of hardware virtualization will henceforth be called a *compartment*.

A hypervisor or Virtual Machine Monitor (VMM) virtualizes a physical platform and enables the execution of isolated compartments. Isolation is basically achieved by granting each compartment access to the CPU, memory and interrupts, whereas the hypervisor stays in control of the MMU. A virtualized platform is illustrated by Figure 9. Each compartment runs its own operating system and set of applications.

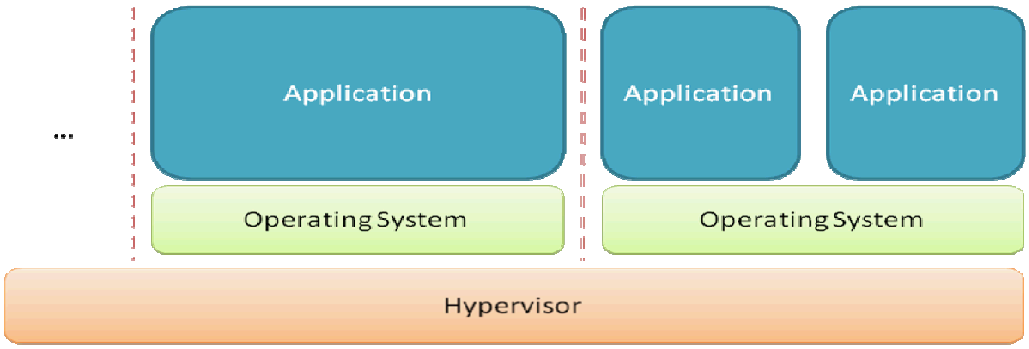


Figure 9: A schematic of a virtualized environment, where a hypervisor manages the operation of several separated virtual machines.

Concerning the SDM concept, a virtualized environment has two beneficial effects: It facilitates platform state measurements by enabling the measurement of a complete compartment and the isolation allows executing services in a trusted state next to less trustworthy compartments.

5.2 *Generic TDS requirements*

A *Trusted Docking Station* is a COTS PC platform with an Infineon TPM and Intel TXT technology and an SDM compatible interface. It must execute a hypervisor that provides

hardware virtualization and compartment isolation. Services which rely on SDM protected keys must be executed in a *trusted state*.

To establish the *trusted state* of such a service it must be executed in a trusted compartment. A trusted compartment is a software image of an operating system which executes a trusted service. The hypervisor measures the compartment using the TPM of the TDS prior to executing the software image. To enable the SDM to verify the measurement, the compartment images must not change between executions. This implies that a trusted compartment must have well defined input output interfaces and that all temporary data must not be part of the compartment image.

5.2.1 Software

In order to be able to run the agent infrastructure the TDS software will need to contain an installation of a recent Java SE, as the agent system will be implemented in Java. It will have to be accompanied with the required security related libraries providing encryption/decryption, signing and signature and certificate verification.

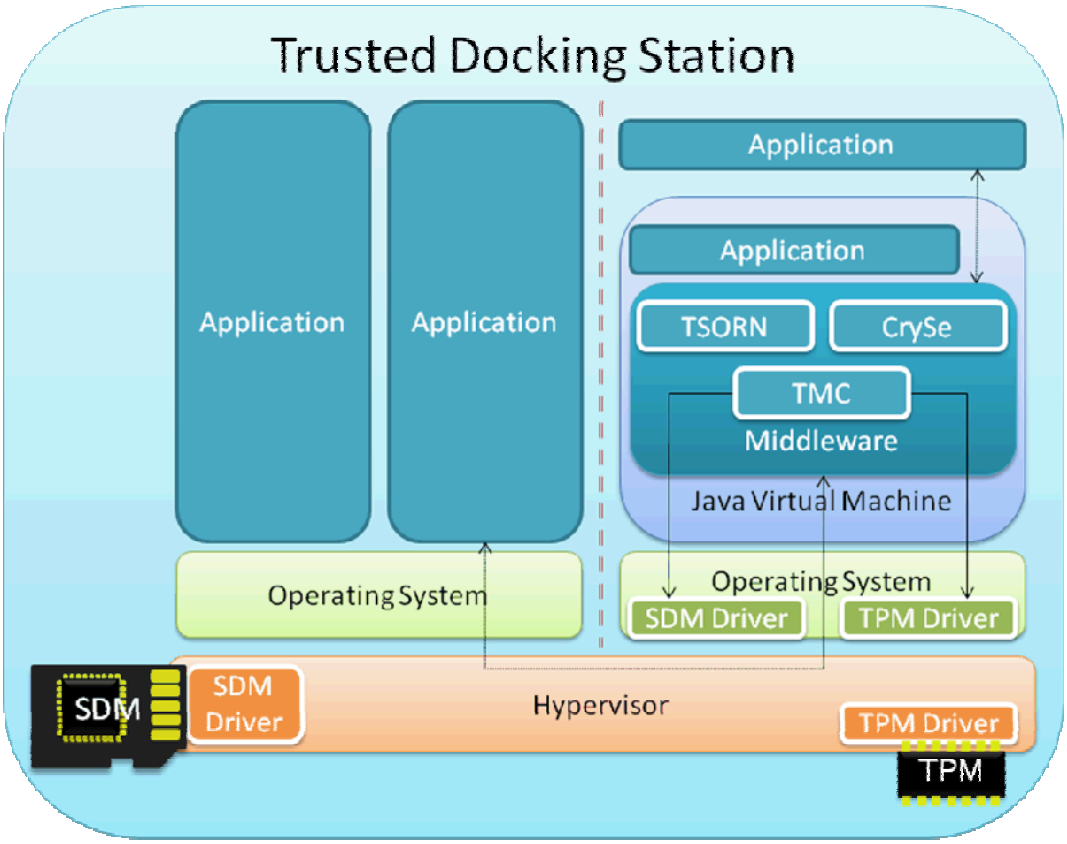
A software driver must be installed that will allow access to the associated SDM device for the purpose of key retrieval by the agent infrastructure application and/or the PTT software. PTT software will also serve as a communication middleware layer for the distributed agent system, providing addressing and secure transfer functionalities. For more information on PTT integration see chapter 6.

5.2.2 HW Specification

A Trusted Docking Station:

- Must support Intel's Trusted eXecution Technology (TXT)
- Must be equipped with an Infineon TPM
- Must have an SDM compatible interface (USB/MicroSD)

5.2.3 Functional Specification and Interfaces



The SDM middleware will provide access to SDM protected keys (TMC). The TSORN part grants access to a network of trusted TDS nodes. The (optional) Cryptographic Services (CrySe) might provide access to basic cryptographic services like encryption with SDM protected keys. If CrySe should be implemented, it will only provide functions like encryption and signing, but not the cryptographic infrastructure like the certificate chains to verify a signature.

What must still be developed is:

- The SDM
- A hypervisor which is capable of using a TPM and measuring the software compartment images prior to execution.
- SDM Middleware

5.3 Specific requirements

TDS for Legacy Systems

TDS for legacy systems is a TDS that supposed to be deployed in a physical proximity of the legacy system hardware. It will run the resource inquiry agents which will be downloaded from the network, which will use a connection to the legacy system for making its queries.

TDS for Operation Centers and Nomadic Stations

This kind of TDS will focus mostly on the interaction with the operator. Therefore it will include graphical interface for entering operator's queries and commands and for displaying agent system requests and results to the operator.

6 Integration with PTT Secure Communication Infrastructure

6.1 Background

The SECRICOM PTT infrastructure is a complex set of physical servers, application services, administrative tools and client applications. Each component of the system has different security requirements. The security requirements even differ further depending on the usage scenarios. There is a different ratio between importance of information protection and performance, ease of use or ease of fast deployment of communication infrastructure. The result is that the PTT infrastructure must be able to work under different circumstances and support both paranoiac level of operation and more relaxed.

The Secure PTT system should support several modes of operation:

- Highly secure operation
- Ease of deployment, maintenance and use

Either way both modes must be able to provide high availability and resilience.

6.2 Integration with Agent system

The Secure Agent Infrastructure (SAI) uses the SECRICOM PTT client components to exchange information between remote Agent components. The SECRICOM PTT provides services for SAI to allow sending and reception of data. The PTT addresses are used to identify source and destination of data. To allow modularity and future extensions of the system the SECRICOM PTT (SPTT) provides its services in the form of open plug-in architecture. The plug-ins will allow SAI to send, receive or display information or request information from a user. The information displayed or requested from user will be structured in the form of HTML, XML forms or other transformation.

6.3 Communication between Agent system and SECRICOM PTT

The agent system will communicate with one or more agent plug-ins for Secure PTT. The communication between the SPTT plug-in and SAI will be realized by IP protocol probably because the SAI is using JRE and SPTT is using C++ and Windows or Symbian operating system. The SAI will use SPTT client components on mobile devices and on SAI servers (SPTT servers wouldn't be directly involved in communication with SAI). The SAI will be executed on TDS and therefore we do not see the need for this cross-process communication (between SAI and SPTT within the same TDS) to be protected. First, the information for other communication node (information that leaves the device) will be already encrypted by keys from SDM and secondly the information to be displayed to user does not need to be protected. The authenticity of system components will be ensured by platform attestation. As mentioned above the binary information to be sent by SPTT to other SAI component will be encrypted by SAI and the SPTT won't be able read it. This might seem as redundant operation (transmit already encrypted content via encrypted communication channel) but this concept supports modularity and higher reusability of

the system and this increased security of information has no noticeable impact on performance or resources.

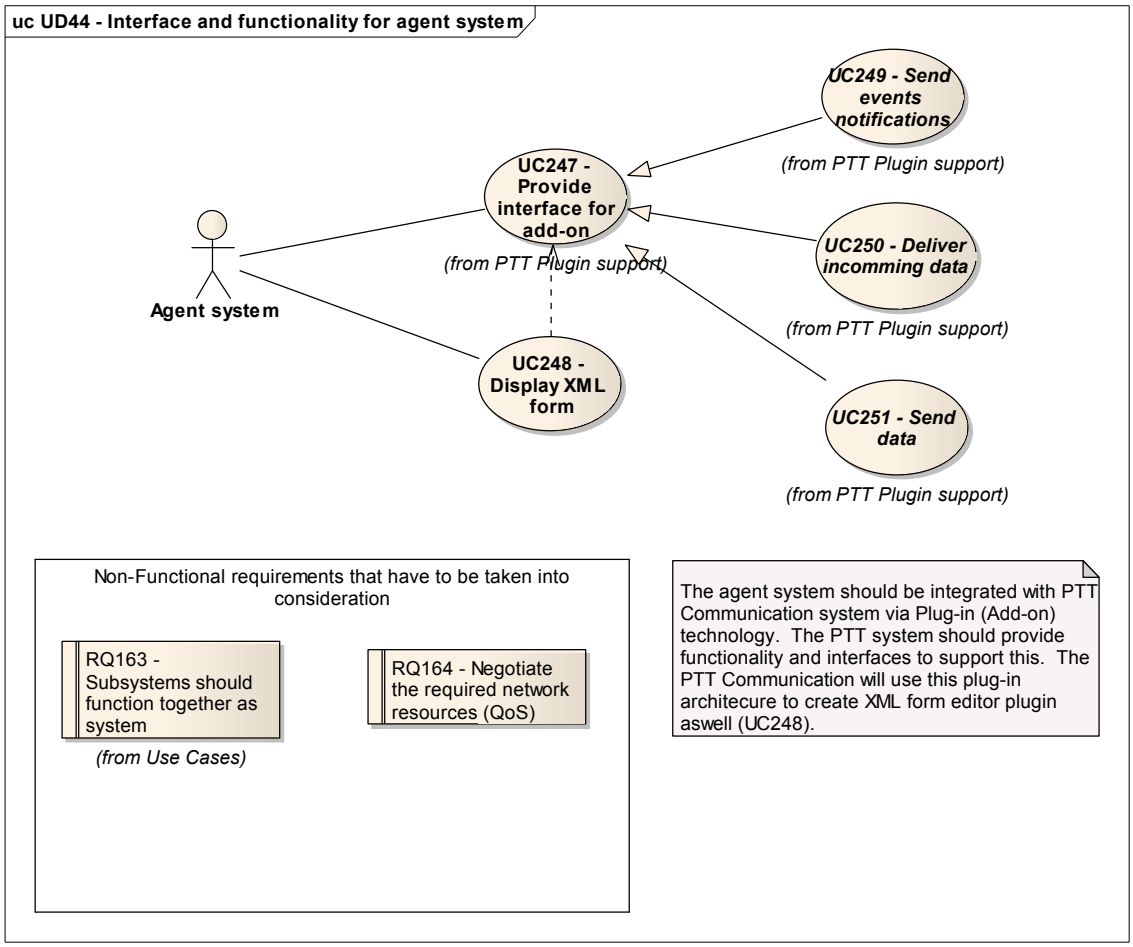


Figure 10: Interface and functionality provided by PTT for Agents

6.4 *Communication between SECRICOM PTT nodes*

The end-to-end communication between PTT nodes is using state of the art security mechanisms to protect the communication line and ensure authenticity of the information. The concept is built upon the fact that the end-users of the communication do not implicitly trust the PTT server. That means the server never has the keys that allow decryption of the communication. The only information the server controls is the signalization (who can speak and when, who is online, offline etc.). The trust between users is realized by PKI and the user of client application must explicitly declare trust in the CA certificate of the authority that issues user's certificates. The CA on the other hand must adhere to predefined policies to ensure the process of issuing certificates is trustworthy.

The SPTT components do not handle issues related to establishing and maintaining IP connectivity between SPTT client and server and relies on lower layers to provide this

functionality. The SPTT system can handle varying level of network quality or bandwidth to limited extend. If the available bandwidth or network latency does not have appropriate parameters some functionality of the SPTT might be limited (e.g.: if there is only 1kbps available the instant audio communication won't be allowed but the text chat, voice mail or other non-instant communication such as sending files should work normally).

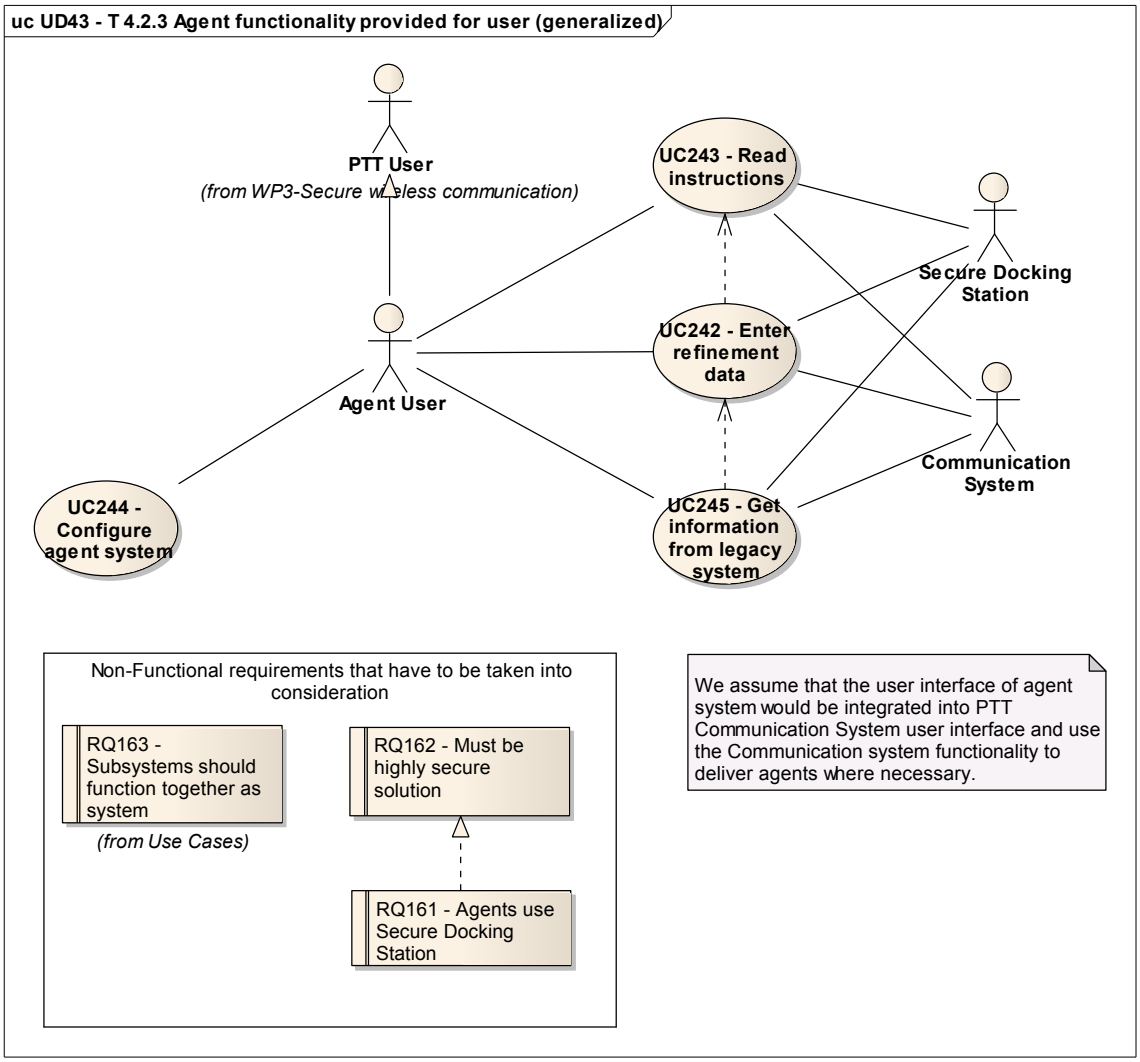


Figure 11: Functionality provided for user interface

6.5 SECRICOM PTT client applications and TDS

The Agents will provide functionality to the end users (emergency responders); the users will use client devices such as, desktop PCs, notebooks, PDAs or mobile phones. To make it easier and user friendly the user interface required for the user interaction related to Agents will be integrated into the user's PTT communication application. The SPTT client application provides the core security mechanisms to ensure authenticity and security of communication and from this point of view this application should be executed in safe environment as well.

As mentioned above, the SPTT system should support two modes of operation paranoiac and more relaxed. From this point of view there will be two versions of client applications, one using pure software implementation of security mechanisms and the other one using dedicated hardware. The hardware could be Java Smart Card with sufficient performance to allow real-time full-duplex audio encryption/decryption. On mobile phones or PDAs the card should preferably have an SD interface. This usage scenario assumes we do not have a mobile device with TMP or SDM functionality and we require high security.

In case the SPTT client application is executed on TDS with SDM where the execution environment is attested, there is no need for dedicated hardware for communication encryption and this can be done in the software. The keys and passwords needed by SPTT could be stored in binary form on SDM directly or it could be stored in the devices memory and only the key that was used to protect the data should be stored on SDM.

6.6 SECRICOM PTT servers and TDS

In the previous paragraphs, the SPTT server was mentioned, but in fact it is more servers, there is different server for each type of communication or services, e.g.: the authentication, signalization, RTP server or CA. Each of those servers could be mirrored or duplicated to provide high availability, load distribution or both. In most cases the performance is the greatest issue because the servers are an un-trusted component (which means they do not handle information in an open form). The one exception is the CA. The CA uses certified Smart Cards or HSM modules to store its keys but the CA requires safe environment and that could be provided by TDS. There are no special requirements on TDS other than to do the platform attestation and provide a way to store passwords or PINs for automated operation of the CA.

7 Security Key Infrastructure Requirements

The secure infrastructure built by the SECRICOM trusted network requires encrypted, integrity protected, and authenticated communications. To ensure this, several types of keys and certificates are required. Figure 12 shows the Secure Docking Station and the necessary credentials which protect communication and where they reside.

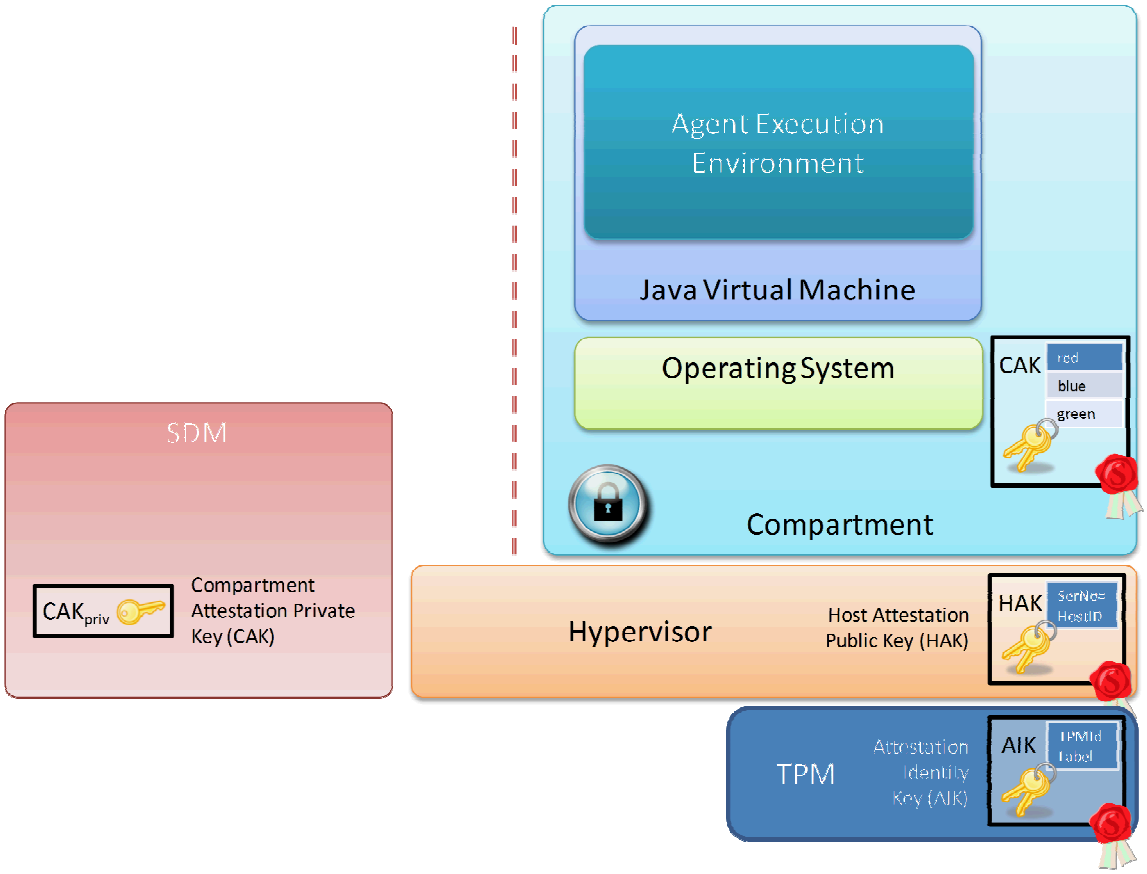


Figure 12: Keys and certificates used in the SDS

7.1 Different kinds of used keys and certificates

In this section the different kinds of necessary keys and certificates are discussed and the requirements for certificates and certificate authorities (CAs) are aligned.

7.1.1 The SDMs own key(s)

The private/public key pair for the SDM (SAK) is a 2048-bit RSA key pair. These keys are stored in a SDMs internal format. They will be used for the key exchange for the observing the communication protocol (cf. section 4.2.1). At present there is no necessity for a transportation format in form of a certificate. To initiate a communication session between the host (TDS) and the SDM, a certificate is not needed. The host implicitly trusts the SDM. Once a use case is found where a Trusted Docking Station must know and decide which SDM it communicates with, this functionality can be easily implemented. In this case the

SDM will be extended to store essential parts (SAK, id, signature value) of its certificate in an SDM specific format. The SDM interface library can then use this information and information available on the host to rebuild the SDM certificate.

7.1.2 Host Attestation Public Key

The Host Attestation Key (HAK) is the counterpart of the SDM's SAK to create the secure communication session between SDM and the host. The Certificates for the HAKs have to be generated by a CA. This must be done during, or before the SDM is prepared for the host communication on the administration host. There are no restrictions to the CA which is issuing the certificates. The SDM itself will never check the revocation information of any kind of certificates, especially not the validity of HAK-certificates. Hence the corresponding CA can be implemented or run without a revocation service. The trust relationship for HAK-certificates will be maintained through the maintenance of the SDMs, which means that the host identifier determines which host is valid. Once a host gets invalid to communicate with the SDM, the host entry must be removed from SDM at the administration host, or the SDM itself must get physically removed. A requirement for the CA of the HAK is that it must issue the HAK certificates with the serial number identical to the host identification number.

7.1.3 Compartment Attestation Certificate

The Compartment Attestation Certificate is used for two purposes

1.) Starting the SDS

During the process of starting the SDS and ensuring that it is in a trusted state, the private key of the Compartment Attestation Certificate is used to decrypt the encrypted compartment before it is loaded and executed. The private Compartment Attestation Key (CAK) may be used to decrypt a compartment decryption key of smaller key size to be more efficient. The private CAK must be protected by the SDM and only released to the SDS if it is in a trusted state. The key material data container is used to store the private key part of the Compartment Attestation Certificate and the hypervisor must initiate a key release process to get the private CAK and decrypt the compartment.

2.) Usage of agent colorization

Agent colorization makes it possible to restrict the execution of agents on certain defined secure Agent Execution Environments (AEEs). That means that specific properties of agents must fit to the AEEs where they should be executed. This enables the selection of which agents may be run on specific Secure Docking Stations connected to Legacy Systems. It is a method to ensure the execution of only one or some categories of agents and guarantees that no other kind(s) of agents can be executed on the specific AEEs. Thus, the usage of connections to the Legacy Systems can be sealed and bound to evaluated agents. This will be achieved through the usage of certificates that are carrying matching values in specific certificate extensions. Agent colorization is also called *agent classification*.

The CA issuing certificates for Compartment Attestation Keys must be able to provide this certificate Extensions as well as a revocation service for the certificates. To enable "offline" revocation checking if an incident happens and the network infrastructure is not available, this service should also provide certificate revocation lists (CRL) which the evaluating services can use to cache. The CRL-service can be used instead of an Online Certificate Status Protocol (OCSP) service or supplement it.

In the process of evaluation and attesting the agents, they will be signed electronically. The issuing CA of these certificates must also be able to provide the certificate extensions for the agent classification. They must be set by the agent evaluation party. Therefore, it must be ensured that only this party may request the certificates for agents. Technically one CA can be used to issue Agent Certificates and Compartment Attestation Certificates. Compartment Attestation Keys must be requested (or eventually generated) by the administration host of the SDM.

7.1.4 Attestation Identity Key

To achieve hardware binding to a platform (a platform's TPM) some properties of that hardware are put into extensions of a special certificate.

The Issuer of Attestation Identity Keys is a so called Privacy CA. The privacy CA is an accessory concept of the TCG and defined by this group. To supply platforms with AIKs a Privacy CA must be employed. A Privacy CA provides privacy to users and applications in a trust-enabled networked environment and nevertheless binds the certificates to a platform (TPM). It confirms that keys are protected by a specification-compliant TPM implementation and thus may be trusted under certain conditions (but without revealing the specific identity of the TPM to a user or application). It is an important aspect, that a Privacy CA requires knowledge of private information of a computer's configuration. Therefore, it must be trusted. The Privacy CA for used issuing Attestation Identity Keys (certificates) belonging to SECRICOM hardware platforms must observe the TCG specification.

8 References

[Bor01] Niklas Borselius: Mobile agent security, Electronics & Communication Engineering Journal, October 2002, Volume 14, no 5, IEE, London, UK, pp 211-218.

[D2.1] Kocis et al: SECRICOM – Analysis of external and internal system requirements. Deliverable report D2.1, the SECRICOM project, February 2009.

[D2.2] O'Neill et al: SECRICOM – Analysis of Crisis Management System Requirements. Deliverable report D2.2, the SECRICOM project, February 2009.

[Fer07] Ana Ferreira, Ricardo Cruz-Correia, Luis Antunes, and David Chadvick: Access control: How can it improve patients' healthcare?, In: Studies in Health Technology and Informatics, 127, June 2007, http://www.cs.kent.ac.uk/pubs/2007/2625/content.pdf

[Gra06] David Grawrock. The Intel Safer Computing Initiative Building Blocks for Trusted Computing. Richard Bowles, 2006.

[Jans01] Wayne Jansen, Tom Karygiannis: Mobile Agent Security – NIST Special Publication 800-19. National Institute of Standards and Technology, Computer Security Division, Gaithersburg, MD 20899.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press, 2001.

9 Glossary

AEE	Agent Execution Environment
AES	Advanced Encryption Standard
API	Application Programming Interface
AP	Agent Platform
AR	Agent Repository
CBC	Cipher Block Chaining
CMCC	Communication Monitoring and Control Centre
COTS	Commercial Off-The-Shelf
CRTM	Core Root of Trust for Measurement
DSAP	Distributed Secure Agent Platform
HIS	Hospital Information System
HPP	Host Platform Providers
HSM	Hardware Security Module
IDA	Information Delivery Agent
IV	Initialization Vector
LIS	Legacy Information System
PCR	Platform Configuration Register
PIN	Personal Identification Number
PMS	Process Management Subsystem
PTT	Push To Talk
RIS	Resource Inquire System
SAI	Secure Agent Infrastructure
SDM	Secure Docking Module
SDS	Secure Docking Station – a combination of TDS and SDM
SHA	Secure Hash Algorithm
SPTT	SECRICOM Push To Talk
TCG	Trusted Computing Group
TDS	Trusted Docking Station
TPM	Trusted Platform Module
TXT	Trusted eXecution Technology
UCA	User Communication Agent
VMM	Virtual Machine Monitor (Hypervisor)

10 Appendix A – Other TDS monitoring parameters

There will be other monitoring parameters in TDS not directly related with the agents, which have been identified as monitoring operational requirements for TDS:

- Version of server and monitoring interface
- Actual state of server (PC)
    - System Resources (CPU utilization, memory usage, load on network connection, available free space on hard drives)
- Actual state of the archivation server
    - number of records in database
    - database size
    - performance – bandwidth
    - number of records per second
    - amount of data per second
    - average time to record data
    - number of opened communication sessions (that are recorded)
- Actual state of PTT server software
    - Version
    - Last restart, reason, who restarted the service
    - List of error states/codes – history
    - Server up-time (how long it is running)
    - Number of connected users
    - Number of active communication sessions
    - Number of open communication slots for users to use (how many more users we can handle)
    - Statistics about data flow (average, min, max...)
- Information about users
    - Name, PTT address, version of client application, type of device used
    - Online time
    - List of users with whom he is able/allowed to communicate
    - List of functionality/features allowed to user (audio only, pictures, text, agents, files...)
    - Role in the system, priority or level in hierarchy (we still do not have clear picture about this – this is just an idea)
    - Presence status (online, offline, away, Do-not-Disturb, In-reconnect, In call...)
    - Actual IP address and type of client device (PC, mobile phone...)
    - List of current communication sessions (+ statistics about those sessions?)
    - Amount of transmitted data
    - Information about user's connection quality (delay, jitter, packet loss)
    - Statistics about user communication
        - Number of reconnects

        - Number of SIP (Signalization Information Protocol) messages per second and total
    - Information indicating if the user has assigned Talk-Burst (means that the user is currently speaking to other users in One-to-many conversation)
- Information about communication session(s)
    - Which active sessions currently exist
    - Type of session (ad-hoc, full-duplex call, chat-room...)
    - Session duration time
    - Current users in the session
    - Session priority
    - State of the session (active, idle...)
    - Statistics
        - Bandwidth used by the session
        - Amount of data transmitted in the session
        - Number of reconnects during session life time
        - List of users with active audio reception (only relevant for PC type of users)