Establishing UE and NE Protection Methods

Security Infrastructure Integration and Re-Evaluation

D23-1.0

Contributors: Fraunhofer Research Institution AISEC
Nokia Siemens Networks Management International GmbH
RWTH Aachen
Cassidian Systems

Editor: Manfred Schäfer (NSN Management International GmbH)
Establishing UE and NE Protection Methods
Security Infrastructure Integration and Re-Evaluation
D23-1.0

About the ASMONIA project

Given their inherent complexity, protecting telecommunication networks from attacks requires the implementation of a multitude of technical and organizational controls. Furthermore, to be fully effective these measures call for the collaboration between different administrative domains such as network operators, manufacturers, service providers, government authorities, and users of the services.

ASMONIA is the acronym for the German name* of a research project that aims to improve the resilience, reliability and security of current and future mobile telecommunication networks. For this purpose the ASMONIA consortium made up of several partners from academia and industry performs a number of research tasks, based on the specific expertise of the individual partners. The project running from September 2011 till May 2013 receives funding from the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF). Various associated partners further contribute on a voluntary basis.

* The full name is "Angriffsanalyse und Schutzkonzepte für MOibilfunkbasierte Netzinfrstrukturen unterstützt durch kooperativen InformationsAustausch" (Attack analysis and security concepts for mobile network infrastructures, supported by collaborative information exchange).

Partners: Cassidian Systems
ERNW Enno Rey Netzwerke GmbH
Fraunhofer Research Institution AISEC
Hochschule Augsburg
Nokia Siemens Networks Management International GmbH
RWTH Aachen

Associated Partners: Federal Agency for Digital Radio of Security Authorities and Organizations (BDBOS)
Federal Office for Information Security (BSI)
Deutsche Telekom AG (DTAG)

For more details about the project please visit www.asmonia.de
# Table of Contents

1 Executive Summary 5

2 Introduction 6

3 Fundamental Aspects for Integration of Protection Methods 7

3.1 Security Infrastructure Architecture (SIA) 7

3.1.1 SIA for NEs 7

3.1.2 SIA for UEs (SW-IP) 12

3.1.2.1 Baseband Attestation 13

3.1.2.1.1 Baseband Hardware Architecture 13

3.1.2.1.2 Baseband Software Components 13

3.1.2.2 Application Attestation 13

3.1.3 Infrastructure Aspects for Malware Detection on UEs 14

3.1.3.1 Deployment Context 14

3.1.3.1.1 Monitoring 15

3.1.3.1.2 Detection and Propagation of Events 15

3.1.3.1.3 Reaction 15

3.1.3.1.4 Message Formats 15

3.1.3.2 UE Related Strategies and Optimizations 17

3.1.3.2.1 Using Kernel Mode 17

3.1.3.2.2 System Call Parameters 17

3.1.3.2.3 Application Monitoring 17

3.2 Trust and SW Management 18

3.2.1 Managing NE 18

3.2.1.1 Multi-stakeholder Governance 19

3.2.2 Managing UE 20

3.3 Messaging 21

3.3.1 SW-IP for NE 21

3.3.2 SW-IP for UE 22

3.3.3 Malware Detection for UE 22

4 Integration into Overall ASMONIA Overall Concept 23

4.1 Refining the FC-IP Cluster for NE 23

4.2 Interfacing of FC-IP functions related to NEs 25

4.2.1 NE towards OAM 25

4.2.1.1 SNMP 28

4.2.1.2 Bulk & File Transfer (SW and Trust Management and Updates) 29

4.2.1.3 TR-069 29

4.2.2 OAM towards FC-MA 34

4.3 Interfacing of FC-IP functions related to UEs 35

4.3.1 Backend Functions associated with Integrity Protection for UEs 36

4.3.1.1 Notation 36

4.3.1.2 Cryptographic Keys 37

4.3.1.3 Concept and Main Ideas 37

4.3.1.4 Integrity Verification of the Baseband Stack 39

4.3.1.5 Generation of authentication Vectors 40

4.3.2 Backend Functions associated with Anomaly Detection (Malware) for UEs 42

4.3.2.1 Communication and Interfaces towards MN 42

4.3.2.2 Malware Collector/Controller Hierarchy 43
1 Executive Summary

This document provides detailed integration concepts for the SW integrity protection and malware detection methods as elaborated in the ASMONIA project, within the so-called functional cluster IP (FC-IP). Essentially, the concepts describe deployment, management and exploitation of these methods, both for NE and UE.

The document is structured into three main sections (Sections 3, 4, 5), which handle UE and NE aspects together, each from its specific perspective and individual need for management, usage and evaluation criteria.

Section 3 first looks into fundamental aspects for integration, e.g., reflecting implications and measures for trust management and deployment, which are directly implied by a specific protection concept or paradigm. The underlying principles ‘to derive anomaly messages’ are shown and appropriate (sample) message structures and formats are provided.

For NE any type of security relevant data (as used in the process flow) is detailed and it is explained how this data has to be treated and protected and by which entity.

For UE the integration of integrity protection mechanisms is described for two use cases. First we consider an attestation scenario for mobile baseband stacks at time of network connect, and second, we consider an attestation of isolated software parts running on the application processor of a Smartphone.

Section 4 refines the internal structure of the FC-IP cluster in order to integrate the individual mechanisms into the ASMONIA specific monitoring and analysis context. Additional processes, entities, and extensions are identified, as needed to deploy the protection methods and to propagate resulting output (which is generated during operation) towards FC-MA.

For NE it is shown that essentially the existing OAM infrastructure can be re-used and (if necessary) data conversions can be handled on a proprietary base. In particular, it becomes evident that for a manufacturer driven approach only minimal interaction with the existing infrastructure is required, not demanding any changes of the standardized 4G network and its communication interfaces.

For UE all mechanisms typically require additional dedicated backend servers connected via TCP/IP. Additionally, for the baseband attestation use case, a minimal modification of the MME is required.

Section 5 is dedicated to describe re-evaluation aspects, confronting the elaborated methods and integration concepts with NE and UE specific evaluation criteria. In a critical view strength and effectiveness of the proposed mechanisms are considered, but also restrictions and trade-offs for implementations are revealed. In a few cases hints are given how to improve or vary the methods, referring to the description in earlier documentation.

Section 6 contains conclusions entirely reconsidering UE and NE views on FC-IP, emphasizing benefits and motivation for applying the FC-IP mechanisms to the favor of improving overall network security in a collaborative information sharing context.
2 Introduction

This document is built on other documentation of the ASMONIA project associated with the functional cluster ‘integrity protection’ (FC-IP). It is particularly referring to [ASMONIA_D21, ASMONIA_D22], which describe security requirements and expectations, state of the art, and finally the concepts and mechanisms as examined and elaborated in the area of ‘integrity protection’ and ‘malware detection on devices’.

While focus of the preceding work has been set on method development for the nodes (network elements NE and user equipment UE) themselves, in this document integration aspects are in the foreground - in order to deploy and to manage the elaborated security mechanisms in the context of 4G compliant mobile network environments. Where needed and applicable, integration concepts consider manufacturer and supplier processes as well, e.g., for trust and SW management, for external analysis for malware samples, and for the modification of devices fostering implementation of appropriate protection mechanisms.

This document provides proposals to exploit protection and detection methods during operation and to propagate resulting messages at network level, towards the so-called functional cluster ‘monitoring and analysis (FC-MA)’ enabling further analysis and support for information sharing.

In addition, the document comprises a re-evaluation of the protection methods, reflecting their effectiveness and correctness with respect to initial requirements, to implementation variants and constraints, as well as to residual risks, which cannot be completely avoided or mitigated.
3 Fundamental Aspects for Integration of Protection Methods

First we consider fundamental aspects for managing and integrating the protection methods (as described in [ASMONIA_D22]) into a mobile network. These aspects are mostly motivated and caused by the methods themselves. Syntax and semantic meaning of the message types themselves are refined or summarized in Section 3.3.

Nevertheless, the principles discussed below, are also closely related to Section 4, which concentrates on ASMONIA integration, starting from the common understanding of FC-IP (in ASMONIA consortium) and in particular, the collection, and propagation of message flows.

3.1 Security Infrastructure Architecture (SIA)

The security infrastructure architecture (SIA) relates to the entities, roles and mechanisms required to securely deploy the protection methods into the mobile network, seen from a manufacturer and operator perspective. Due to their specific nature there is no common solution for NEs and UEs, and even for UEs the malware related methods must be considered separately from SW integrity protection.

3.1.1 SIA for NEs

In this section we refine the SIA required for network elements, following the protection concepts introduced in [ASMONIA_D22]. First we detail the generic SW-IP system (see [ASMONIA_D22], Section 3.2) for NEs, focusing on methods as required for protected SW and data, based on asymmetric cryptography. Mechanisms relying on pure hash values, we do not further consider in this document (in NE context).

![Security Infrastructure Architecture for NEs](image)

**Figure 1: Security Infrastructure Architecture for NEs**

We start from component creation and follow the block-arrows, tracking a typical material (HW) or data (protected data, sensitive data) flow during product life cycle. While the individual domains and their assigned tasks and duties already have been described in [ASMONIA_D22], here we concentrate on the processing and transmission of objects along this path. According to the individual production, delivery, and usage steps we distinguish...
(1) Protected components, created by TTPs (Trusted Third Party) and delivered to the NE manufacturer (vendor). Integrity of an NE requires that all its components (HW and SW) are integer. Component generation involves testing and checking that a component is exactly implementing its specification and does not contain vulnerabilities or backdoors. While residual risks will remain, to a huge extend this can be assured applying technical (vulnerability checking, testing) or organizational controls (audits, reviews, contracts). Regarding HW we assume that this is assured applying secure production and delivery processes and acceptance tests (note, that in ASMONIA we do consider HW integrity issues, except for some insights into anti-tampering, see [ASMOINIA_D22], Section 3.7.3).

Regarding SW a suitable data protection mechanisms shall be applied, securing data during transmission and storage. Concrete security controls are a matter of mutual, contractually or otherwise established trust relations between TTP and manufacturer:

a. If there is no contractual relationship (e.g., if TTP is an open source community), responsibilities for testing/checking may be different, possibly imposing additional efforts at vendor side (e.g., for testing). However, in many cases the recipient simply applies the integrity protection mechanisms provided by the community (e.g., as it is done with signed packages) and does not further influence these.

b. If the TTP has a contractual relationship with the manufacturer, the manufacturer may rely on the security controls applied by the TTP or may negotiate individual mechanisms, e.g., by aligning the TTPs integrity controls (underlying cryptography, production process) with its own expectations (security policies) and controls for validation (to give an example, the manufacturer may accept the TTP’s root certificate when validating TTP signed components, based on an agreed signed object format). In addition (based on contractual agreements on trust and quality assurance) he may reduce its checking efforts on simply applying cryptographic validation using the agreed protection method.

(2) Validated components are used to create a product. Validation is executed in accordance with the mechanisms explained above. Once the components are validated (e.g., a SW library or a complete application) they can be further processed, i.e., just used for linking or even modified and adapted and then used as a fundament for own applications.

We require that subsequent manufacturing processes are done in a trusted environment, so that during the product creation no integrity mechanisms are executed, but extensions may be implemented into the product itself (e.g., for self-validation during an installation process). Certainly, such trusted environment involves a bunch of measures and security controls (e.g., on identity and access management to prevent unauthorized persons from misuse), which may be very specific to the manufacturer, e.g., using trusted tool chains (compiler, linker, SW management system) and secure processes in development environment, assuring that only SW and data is signed, which has been tested, released (product released) and approved before.

(3) Data required for signing is used during protection processes. Such data involves

a. the data to be signed or a hash thereof
b. a signing request including (a),
c. the signed object
d. the secrets required for signing, i.e., a signing key.
At first view only the signing key looks like a threatened secret, but actually also (b) and implicitly (a) are highly threatened, as it is trusted data (but are not secret) pointing to the objects that ought to be protected. It has to be assured that these objects actually are approved for signing. Concluding, the entire signing process is sensitive to attacks and need protection, at least requiring accurate Identity (authentication) and Access Management (authorization) to prevent illegal persons from misusing the signing process (e.g., pushing malicious objects for signing). It is a matter of the local working conditions, whether and which further security controls are required. Two factor authentication and four-eyes-principles in approval workflow are practicable examples.

(4) Data required for validation is transmitted to the validating recipient (depending on the used case this can be either an external system or the target system itself, compare [ASMONIA_D22], Section 3.2.2.). We distinguish transmission of material (HW, printed media, storage media) and the data itself, which can be part of the material (e.g., burnt onto a CD, in USB stick, or as firmware, shipped together with HW) or also transmitted independently via electronic media (e.g., in IP data stream). As soon as we talk about (integrity) protected data (signed objects) this is end-to-end secured (regarding unauthorized modification), independent from other security conditions during transmission. The only relevant attacks would be to replay (e.g., re-install vulnerable deprecated, but legally signed data), suppress, or delay transmission, but this could be controlled by other means (e.g., versioning, revocation or confirmation of recipient).

Sensitive data is the only data that needs protection, be it as part of transmitted material or as separate data. Using a PKI based integrity protection scheme the sensitive data essentially include

a. the Root CA certificates (trust anchors)

b. revocation information for signed objects/data or even associated intermediate certificates and possibly cross-certificates

c. data authorizing to exchange, add or remove trust anchors

Relevant attacks are exchange (a), suppression, deletion or delay (b, c), assuming that b and c itself would be integrity protected (validated against a valid trust anchors or secured by administrative means). Thus, the fundamental root of security is the trust anchor, which must be ‘immutable’ associated with the validating system, i.e., only authorized persons or processes shall be able to overwrite an installed trust anchor.

In Figure 2 an example is shown for validation paths including cross certification. Note that governance for cross certification behooves to the CA, which is stored as the protected trust anchor in a validation system; in this case the root CA A would issue a new certificate containing the public key and the distinguished name of B as the origin issuer by that extending the validation path up to the trust anchor.
(5) After delivery any material will be deployed, and activated (HW), installed, or otherwise used (data read from moveable media). For sensitive data specific security controls must be applied, but in many cases it could be shipped with the product as part of the hardware or as protected firmware, respectively. In some cases (e.g., if only SW – as the entire product - is shipped to a customer), the (initial) treatment of the trust anchors need special security controls. This can either be accomplished by administrative means (customer care service) or by administration of a (trusted) operator itself. Protected data must be validated before or even during use, according to the use case applied for integrity protection (compare [ASMONIA_D21], Section 3.2.3.1). Depending on use case and on implementation any shipped data will be given either to the product space (5a), or into a NE management system (5c), or into a separate infrastructure (5b), which may be required in cases, where external validation is indispensable (e.g., if the target system is SWIP unaware).

(6) In many cases validation of protected data is done by the product itself (6a) or by the OAM system (6c), but it could also be part of an operator’s obligations (6b), e.g., when receiving protected data via moveable media he might validate this in a separate system before it is installed into a NE or given to an OAM system. Also for protected SW and data stored on repositories, parts of an existing infrastructure could be used and extended accordingly. It is then up to an operator to setup proper validation processes, relying on the correct trust anchor and essentially also based on manufacturer provided validation components. Only in a few situations an operator may require small investments into additional infrastructure. Regarding sensitive data, the relevant attacks are as listed in (4) and in addition the validation software has to be reliable (integer), usually implying additional security controls.

(7) Material given to service or repair (S&R) entities has to be treated according to the manufacturers security policies and using same (or compatible) cryptographic integrity controls. Usually only HW is given to S&R, but implicitly it contains also sensitive and protected data. During delivery it has to be assured that sensitive data cannot be modified (in particular, when given back to operator). S&R entities have to be trusted, acting as part of the manufacturer or as a subcontracted trusted third party. From SW-IP point of view S&R processes must be aligned with the product creation and validation processes, following the same or compatible signing and validation schemes and security policies. Regarding signing, preferably the trust anchor could be the manufacturer’s root CA certificate and S&R entities could possess either a
signing infrastructure, which is part of the manufacturers PKI hierarchy (sub-CA + signing service) or is realized as a signing service authorized via a valid signing certificate (issued by the manufacturers PKI), associated with valid private keys for the signing process. It is also feasible that S&R entities use the manufacturers signing service (e.g., for modified SW and data) through a protected channel. In a variant it is also possible (but most likely not economic and very difficult to manage over the entire product life cycle) that products implement additional root CA certificates (‘second’ trust anchor) of authorized S&R entities and these use an independent signing infrastructure, which is dedicated for S&R changes. Moreover, solutions using bridging CAs are imaginable. However, as regarding trust management a manufacturer governed solution seems to be the best approach, only the first mentioned variants (authorized sub-CA or signing service, or using manufacturers signing service remotely) may be applied in practice. Note that delivery of the repaired units back to the customer (MNO) follows similar principles as described in (4) (but may possibly be associated with a separate trust anchor).

(8) In accordance with (7), security data and agreements between manufacturer and repair services (S&R) have to be aligned very carefully, not to enable security leaks. In any case a repair service has to be trusted as during repair it has full access to the HW and could manipulate it in any arbitrary way. Thus, there must be organizational controls, which may be technically or even procedurally enforced (at least, by trust and contract). In many cases the S&R may use signed material provided from the manufacturer, e.g., a patch or SW update. But in a few cases a S&R may also make adaptations to SW or data (for example when individually calibrating a new configuration for a module due to new or repaired ‘analogue’ HW) requiring use of a delegated or remote signing service, which is fully compatible with the manufacturers signing policies and principles. Such agreements may be product or release specific.

(9) See (10)

(10) Results from validation may be collected from NE management systems (9) or from external validators (10b) reporting to NE systems or to the ASMONIA interfaces directly (10a). In addition (10b) may also be used to transfer protected data into OAM systems. Via these interfaces also control mechanism could be applied, as described in Section 3.2.1. Validation results and their treatment will be further described in Section 3.3.

(11) Collaboration data is the information, which is shared between operators as a result of analysis and evaluation by the ASMONIA system. Regarding NE Integrity protection, such data is generated based on a collection and aggregation of data coming from individual NEs and other sources, such as OAM systems and external validators, e.g., a SW repository or SW download system (if applied).

As evident from Figure 1 the SIA for NEs partially results from the methods being selected and developed to assure integrity protection and partially it results from the overall ASMONIA approach, which may be seen as an overlay network interfacing with extensions and functions specific to ASMONIA. It also shows that essential parts of the SIA are realized within the operator network; however, these do not largely influence the standardized operator infrastructure. Nevertheless, the SIA has a few (proprietary) impacts on the NW management and OAM systems. While NW management system anyway are designed to cooperate with NEs from different manufactures, we assume that FC-IP extensions at NE-level to a large extent can be integrated into 4G products themselves (NE) or via proprietary parts of OAM and SW management (repositories) systems. However, for aggregating and propagating
validation messages, common interfaces and functions will be necessary, capable to operate in multi-vendor network environments.

Additional local protection measures:

Supporting the protection given by (product-integrated) SW-IP and recommended hardening, additional protection and analysis options could be provided through intrusion detection and log-file analysis offered by Host Based Intrusion Detection Systems (HIDS), such as OSSEC\(^1\). In contrast to SW-IP for NE, usually HIDS are administered \textit{locally}, not (only) based on cryptography, but in particular on an evaluation of conspicuous behavioral events occurring during operation. Actually, HIDS are generic approaches (running out of the box), but must be closely aligned with specific properties, type, version, expected behavior, and configuration of individual monitored systems (i.e., system dependent rules and evaluation logic may be needed). While HIDS techniques are not directly in scope of ‘integrity protection’ (as considered in ASMONIA) these also aim to ensure a system’s reliability and integrity. Thus, deployment of HIDS can contribute positively by gathering additional “anomaly” information per system, which can be aggregated and evaluated at network level.

As HIDS system usually are realized via client-server architectures ‘on top’ of a specific system landscape (including different products of several vendors), in this case the optimal way for integration should be decided by the network operator. The associated infrastructure usually requires separate HIDS servers and protocols, but these may be integrated with the OAM systems (if supported by NE manufacturer).

3.1.2 SIA for UEs (SW-IP)

Mobile devices in a 3G or 4G network are a very heterogeneous mass of devices. However, individual devices always consist of two main (hardware) components: the baseband (processor) and the application (processor). The following figure shows such common multi-CPU architecture for smart phones and 3G USB modems, where the processors communicate via a serial line or shared memory.

![Common device architecture](http://www.ossec.net/)

In the following, we focus on two use cases for integrity protection on mobile devices. First, an attestation of the baseband at network connect and, second, the verification of software components running on the application processor in the course of a remote transaction, e.g., a payment or banking transaction or remote access to corporate data via a VPN or similar mechanisms. The first scenario was first published in [WWS2012] and a research paper dealing with the second scenario is currently under review. Both mechanisms require an additional server in the backend accessible via TCP/IP.

\(^1\) http://www.ossec.net/
3.1.2.1 Baseband Attestation

In the paper Attestation of Mobile Baseband Stacks [WWS2012], we presented a hardware-based attestation - more specifically, a concept, an architecture, and a protocol - for mobile baseband stacks. Such attestation enables a mobile device to efficiently prove its trustworthiness towards the network without the need for expensive asymmetric cryptography. Instead, symmetric operations transfer the result of the attestation from the prover to the verifier. Based on the attestation, the mobile network can grant (or restrict) the access to critical core components. As a result, the risk and the potential damage of attacks on the network from devices with a compromised baseband stack can be limited. It even enables the network to enforce a certain baseband version, which prevents attacks that exploit vulnerabilities in a (prior version of the) baseband stack in order to attack the network. Besides an additional server in the backend, this mechanism also requires minimal modifications to the MME, as described later in this deliverable (see Section 4.3). Furthermore, we assume the following baseband hardware architecture and software components.

3.1.2.1.1 Baseband Hardware Architecture

The baseband hardware in mobile devices usually consists of the following parts: RF front end, analog baseband, and digital baseband consisting of a DSP and an ARM SoC. In this deliverable, only the ARM SoC (baseband processor) is considered.

To the baseband processor, an exchangeable USIM is connected, which is a smart card, issued by the provider. The USIM holds in its ROM an operating system, and the security algorithms for authentication and key generation. In its EEPROM, it stores specific identity information, namely the IMSI and TMSI as well as a secret $K_i$, which is shared with the provider.

However, today’s mobile devices completely lack a comparable secure element to identify the device itself. The existing device-unique IMEI, for instance, is not stored securely, hence needs to be considered untrusted. For our concept, we therefore propose to extend the hardware architecture with a Mobile Trusted Module (MTM), see [TCG_MTM]. In contrast to existing solutions, we propose to directly connect the MTM to the baseband processor.

3.1.2.1.2 Baseband Software Components

Today’s baseband software is usually a small real-time operating system which is responsible for parts of layer 1 (hardware specific physical layer) and everything above. For layer 2 and 3, it provides a hardware independent software stack with nested implementations of all 2G/3G/4G layers.

For the sake of simplicity, we divide the baseband in the following two parts: the baseband binary (B) and baseband information (BI) including device- or operator-specific configuration data. Although there might exist other binaries, such as fail-safe or backup binaries, only one baseband stack is running on the baseband processor. A boot loader, which is usually very small, loads the baseband binary from memory. We assume that the loader is stored in ROM to securely boot the baseband stack and authenticate updates that need to be cryptographically signed by the baseband vendor (BV).

For our protocol, we extend the boot loader and the baseband stack with the functionality to communicate with the MTM. Additionally, the USIM software now executes critical parts of the minimal TSS internally.

3.1.2.2 Application Attestation

The attestation of software components running on the application processor is based on previous ASMONIA results. In [Horsch2012] a software-based trust anchor for ARM Cortex
application processors was presented. This mechanism builds the foundation for a software-based secure boot. In the last step of this boot process an Android kernel is started, which is extended with operating-system-level virtualization features as described in [Wessel2013]. This mechanism allows the isolation of trusted and untrusted userland components running on the application processor, e.g., to isolate pin entry dialogs or components for banking or payment scenarios. In this case, only the components belonging to the trusted computing base are attested. Since the backend implementation in this case is completely independent from the operator and only requires a simple TCP/IP connection to a single dedicated backend server, a more detailed description of the functionality of the server is out of scope for this deliverable.

3.1.3 Infrastructure Aspects for Malware Detection on UEs

This section first describes the management context for malware detection on UEs, messages associated with malware detection events, as well as optimizations on UE level. In Section 4.3.2 later we expose, how malware detection on UEs is integrated into the mobile network and into the ASMONIA overlay architecture.

3.1.3.1 Deployment Context

The anomaly based malware detection as detailed in [ASMONIA_D22] is envisioned to be deployed on a variety of UEs in the MNO domain. Figure 4 depicts an abstract view of the mechanisms deployed on an individual UE and their communication processes towards external entities. In this context we introduce the notion of a ‘Regional Malware Collector/Controller’ (RMWCC) as the first instance communicating with an UE. RMWCCs collect the events generated by UEs and manage MW detection mechanisms according to applied strategies.

![Figure 4: UE Anomaly Monitoring](image_url)
3.1.3.1.1 Monitoring

Within an UE the individual user applications (apps) running on the UE are each monitored via the detection mechanism according to the specified observation model, i.e., the logical coherence of system calls. The detection mechanism holds individual models for each app representing their respective benign behavior of their system calls. At runtime, the detection mechanism captures system call traces of running apps, e.g., using ‘strace’ [STR] on Android based UEs, or other, preferably kernel based mechanisms on other UE platform such as the iPhone. The detection models for the individual apps can be updated by improved models collected across multiple UEs with the same properties. The models can be supplied to the UE detection mechanism by the associated Regional Malware Collector/Controller (RMWCC, see Section 4.3.2.2).

3.1.3.1.2 Detection and Propagation of Events

Once the detection mechanism has determined a sufficient deviation from the specified benign behavior a detection event is generated. Information about such events may include important data such as:

- UE Model
- Vendor
- IMEI of UE
- OS Version
- Name and version of application
- App Meta Information
  - Size
  - Author
  - Date of installation
  - Origin
- Manifest of application
- References to event specific data (e.g., further logs, sys-traces etc.)

The event information is first sent to a Regional MWCC (cf. Section 4.3.2.2) which is responsible for the UE in question. The RMWCC can query the Top Level MWCC (TMWCC, cf. Section 4.3.2.2) to determine whether the reported incident resembles a new unique event. That is done, for instance, if the specific event information constellation has not been captured before. If so, the RMWCC obtains the actual malware sample, e.g., the application binary from the UE and relays it back to the TMWCC which may further analyze the sample, e.g., by other external, non-MNO based mechanisms.

3.1.3.1.3 Reaction

Once a specific threshold (amount of malware; seriousness of malware; possible impact on other UEs; threat level of UE,...) is crossed for a specific UE, a reactive measure can be employed by the MNO, which is responsible for this device. Depending on how much knowledge about the malware in question is available, the MNO may decide to block all or only malware related traffic originating from this UE. Notifying the owner of the infected device may also be desirable, e.g., in order to communicate measures for disinfecting the UE, possibly even by services supplied by the MNO.

3.1.3.1.4 Message Formats

For the messages exchanged between the RMWCC and the UE we propose the BSON [BSO] format. Similarly to the popular JSON format, BSON is a serialization format, which is binary and more efficient.
The key characteristics of BSON are:

- Lightweight in terms of representative overhead, thus making it a good candidate to be used over the network in this scenario.
- Traversable for database usage, especially when used with MongoDB [MDB]. This is useful as the event data should be easily accessible to analysts doing complex queries on the data set.
- Efficient in terms of the encoding which has been proven for many languages (e.g., Erlang, Haskell, Lisp, Python3 or Smalltalk) as it uses C data types.

The generation of BSON encoded data is quick, and additionally allows for human readability in the decoded format.

The following shows an example of representing an event as decoded-BSON:

```json
{
  "Vendor": "T-Mobile",
  "IMEI": 49064940314172,
  "Model": "Galaxy SII",
  "OsVersion": "Android 4.0.2",
  "Application": "Like a Sir",
  "Manifest": "blob",
  "app_meta": {
    "Origin": "http://hotnewapps.biz",
    "Installation_Date": 121212,
    "Author": "dudexyz",
    "Size": 203
  },
  "timestamp": "2012-12-17T15:45:06Z"
}
```

The event data is simply transferred from the UE to the responsible RMWCC via a network connection, e.g., using TCP as the transport protocol. Security measures that may be required on the connection between UE and RMWCC may either be employed at the transport layer, e.g., using TLS, or by directly applying the measures on the BSON encoded event message. It could simply be integrity protected and encrypted. However, this would still require the definition of a wire-protocol between the two endpoints, e.g., state keeping etc..
The message format between RMWCC and TMWCC may also use BSON on the connection via the S11 interface.

3.1.3.2 UE Related Strategies and Optimizations
This section considers strategies and improvements related to UE based anomaly detection.

3.1.3.2.1 Using Kernel Mode
In the proof of concept setup we developed for the Android platform, we decided to use ‘strace’ to monitor the individual applications, i.e., the respective UNIX process they are running in. The monitoring is then effectively controlled by another user space application. The rationale behind using this user space tool was its general ease of use compared to developing a custom tailored Loadable Kernel Module (LKM).

Even though LKMs are significantly harder to develop and especially to debug, a production mode setup should be implemented as a LKM. Using such an approach enables vendors (and MNOs) to tightly integrate the anomaly detection mechanism into the UE. Unloading the LKM or its full removal becomes significantly more difficult for users and in particular for malicious applications. Also, the security parameters used for the connection between the UE and the RMWCC can be embedded more securely.

3.1.3.2.2 System Call Parameters
Currently the PoC only considers the logical coherence of system calls emitted by the UNIX process an Android app is running in is monitored and compared to a pre-trained model. Further data such as system call parameters and return values are recorded, but not included in the modeling and detection.

Research indicates that extending the mechanism in this direction may significantly improve the detection quality and discrimination power of the models [MVV06, MMZ10].

3.1.3.2.3 Application Monitoring
Applications for various purposes running on the UEs have been the killer feature for the success of Smart Phones all across. Users typically install and use a varying set of special purpose apps from different sources and developers. Such sources and the developers themselves may already enable users to heuristically filter which applications the anomaly detection should monitor.

For instances, heuristic filtering could be applied to classify applications:

- Apps form China or other countries which are known to distribute a significant amount of malicious apps.
- Apps form 3rd party markets. This may also be classified further if relevant knowledge to discriminate the market is available, e.g., Amazon Marketplace vs. AppBrain.
- Based on the trustworthiness of the developer, e.g., Google. Again, this may be classified further based on knowledge of the developer and his track record of apps.
- App markets already sort apps based categories, whether they are free or not, popularity, etc.. This could also be used to decide which apps are most relevant to be monitored.
• The set of permissions may also allow pre-selecting, which apps should be monitored, e.g., an app using a specified set of permissions that are considered to be critical depending on the context the UE is used in.

Monitoring applications which are more likely to be malware related can significantly improve the efficiency of the detection mechanism running on the UE.

Filtering apps could go as far as allowing a white-list of applications, which will not be monitored at all. This decision could be made based on a combination the properties of the application, e.g., its author or its installation origin.

3.2 Trust and SW Management

In this section we examine the mechanisms for trust and SW management as induced by a specific method or type of target system.

3.2.1 Managing NE

In NE case trust and SW management to a large extend can be integrated into the product and OAM systems and thus, are determined by the manufacturer (or ‘stakeholders’, respectively). The selected methods in most cases enable and support remote management through provision of signed data and SW.

Decisions on product life cycle (e.g., which SW version is installed), still can be determined by an MNO on local needs, except in cases where this contradicts to higher order security controls and agreed policies (e.g., revoked SW versions, which should no longer be used, or mandatory updates).

Deployment of PKI / signature based SW-IP can be done smoothly during product evolution, which in some cases has to go hand-in-hand with supporting OAM functions. In many situations new SW will ‘simply contain’ the SW-IP functionality (e.g., a new SW version may contain runtime-SW-IP functions in kernel). However, in some cases FW or even HW updates may be necessary (e.g., if high security levels or physical protection is needed). But once roots of trust are established, SW-IP can be controlled either autonomously by the system itself (self-validation during boot, installation, fall-backs, and runtime protection) or remotely (signed SW updates, trust anchor management, revocation, etc.). Initial deployment scenarios usually require trusted operations, which may be executed or controlled by trustworthy service(s) or administrative processes. All SW-IP scenarios assume that an MNO is a trusted instance in alignment with an agreed governance scheme. Nevertheless, the mechanisms provided also prevent from insider-risks in MNO infrastructure (as they may happen in every organization), given they have been properly set-up.

Despite far reaching system support and autonomy, exceptional situations cannot be excluded, where a system cannot recover from an incident and must be repaired or reset by manual operation or dedicated recovery procedures. Essentially, this depends on strength of technical implementation and may happen, if an attacker is able to outfox chosen protection mechanisms by physical access (and sufficiently strong malicious manipulation) or by subtle SW attacks, which surmount the hurdles erected by the hardening methods (it is assumed, that the proposed cryptographic mechanisms themselves cannot be cracked and are future-proof for next ~20 years and probably more, seen from today’s perspective).

As SW-IP itself does not involve local secrets, end-of-product-life (which is not induced by the selected mechanisms) does not require particular actions (such as deleting or revoking a local key). But note that additional security (e.g., authentication mechanisms, local signing, etc., involving private keys that must be prevented from cloning) may imply separate treatment.
However, if we interpret ‘end-of-life’ so that a product is re-deployed for another environment (e.g., change of MNO, sold to other company) appropriate functions must be provided, in particular those allowing change of root CA certificate and associated validation parameters and revocation mechanisms. It is obvious that such functions must be very well protected to prevent from abuse. Compare Section 3.2.1.1 and [ASMONIA_D22], Section 3.7.2.7, for proposals on trust anchor exchange, which may be used and extended to also support re-deployment scenarios.

3.2.1.1 Multi-stakeholder Governance

Multi-stakeholder concepts for NEs have already been discussed in [ASMONIA_D22], Section 3.2.1. They include usage of mechanisms like

- **Second Trust Anchor**
  Such approach could serve either as remediation mechanism against compromise of Root CA or to allow second root CA in parallel, so that SW and data can come from a second trusted source.

- **Hierarchical Delegation**
  such approach could be applied to TTP, as described in Section 3.1.1

- **Cross Certification**
  would also allow control over signed objects by more than one party, see example in Figure 2

- **Bridge CA**
  Is a flexible solution regarding mutual PKI integration, but requires multi-party agreements on selecting and accepting Bridge CAs and in particular the related policies; one specific problem could be the validity periods of the issued certificates (but also the bridge CA root certificate itself), which have to be aligned with the product life-time (that may be 20 years or more for some NEs).

- **Virtualization concepts**
  would allow to create different, but separated trust domains inside a target system

It must be assured that governance principles are clearly defined and followed, so that roots of trust cannot be abused or bypassed from one of the involved parties. For instance, when using virtualization the owner (controlling the Trust Anchor in the HV domain) of the hypervisor might be able to access trust anchors in Virtual machines (VM), while vice versa this seems to be far more complicated (and actually should be prevented…).

Each multi-stakeholder concept comes along with specific requirements and also risks regarding reliability and responsibility for trust management concepts, which have to be accurately considered. For instance, when using a second trust anchor it’s meaning and usage has to be aligned and controlled very carefully, otherwise protection paradigms could be mutually overwritten or disobeyed. This is in particular a problem if trust anchors belong to different PKI domains.

In particular for HeNBs it is imaginable and supported that an MNO’s trust anchor (or MNO selected trust anchor) is also installed, following 3GPP TS 33.320 in the context of the TR.069 CPE management protocol\(^2\). Regarding SW download in TS 33.320 (Section 8.4) it is stated:

".. The file shall use the signed package format according to TR-069 [15]."

\(^2\) See http://www.broadband-forum.org/technical/download/TR-069_Amendment-2.pdf
The SignedData object in the signed package shall contain at least one signature provided by a software signing entity, with certificate issued by an operator trusted CA.

The TrE shall use a public key issued by an operator trusted CA to verify the software signing entity certificate and the signature(s) in the SignedData object. If the verification fails, the H(e)NB shall delete and not install the software that failed the verification. All root certificates used for this purpose shall be stored in the TrE.

NOTE 2: Notification of this failure is according to procedures in TR-069 [15].

The signed package shall also contain the trusted reference values needed for the software integrity checks performed during secure boot.

NOTE 3: TR-069 [15] supports multiple signatures which can be used for the purpose of supporting different hash algorithms.

It is a matter of interpretation if cross-certification would also solve the above stated requirements; technically it would be a feasible solution as well, organizationally this may be difficult to handle as PKI-related contracts and agreements (e.g., on certificate policies) need to be negotiated between several companies.

For other NEs than HeNBs there is no specific requirement known in 3GPP context and it remains a business decision, whether to agree on multi-stakeholder concepts or to exclude these. Compare [ASMONIA_D22] for related issues.

Note that trust management concepts must support the long-lifetime of NEs and multi-party governance may be much more complicated regarding revocation or trust anchor exchange as compared to the single-PKI case (compare [ASMONIA_D22], Sections 3.7.2.7 and 3.7.2.8). In case of revocation conflicts can occur, which need to be resolved: For instance, one signing authority wishes to revoke a signed but vulnerable SW package (e.g., providing shared libraries). However, it ought to remain installed as long as a second authority’s signature over a dependent package remains valid or both packages must be invalidated. Similar conflicts may arise regarding the validity/expiration of plurally signed objects. For such situations it would be necessary to cross-check the signed objects (i.e., the complete validation context) and to resolve the detected conflicts in accordance with pre-defined and mutually agreed rules, such as <invalidate a signed object, if at least one of the signing parties have revoked it or revocation is implied by dependent, invalid objects>, whereas a signed object could remain valid if a signing key of an individual signing authority just has been expired and was renewed. This is a matter of consistent validations policies, which e.g., must be defined in dependence from the certificate and signing policies of the issuing PKIs and/or signing services.

3.2.2 Managing UE

Mobile devices in a 3G or 4G network constitute a very heterogeneous mass of devices including conventional feature phones with proprietary firmware, smartphones, tablets, notebooks with various operating systems or even devices in machine to machine use cases, like traffic lights. And of course, typically, the devices belong to their owners and not to the network operator. So, an overall managing of all devices is not possible. Instead, the management depends on the concrete use case and implementation.

In general, our mechanisms are based on existing software delivery mechanisms. If necessary, these are extended with hashes and signatures. For remote attestation scenarios, the hashes (e.g., for software packets or firmware images) are also stored in the backend in a whitelist. Theses whitelists optionally may include further information like software versions etc.
3.3 Messaging

To ease finding, this section collects all information on ‘messages’ as provided by SW-IP or MWD methods. It either references or refines the definitions given in other documents or it links to other sections where specific information is given.

3.3.1 SW-IP for NE

At informative level in [ASMONIA_D22], in Section 3.6, a generic format for NE-related messages has been introduced. The intention behind was to give an idea what type of messages can occur and which semantics may be behind. Nevertheless, as discussed later in this document (e.g., see Section 4), this is a proprietary area and message syntax and semantics is up to the needs of a specific product or manufacturer. Adapting such messages to a common ASMONIA wide context can be accomplished by appropriate conversion functions, as explained in Section 4.2.2.

In Figure 5 refinements of the generic format given in [ASMONIA_D22] are proposed, meant to be used as an exemplary template for implementation. Major changes address clearer message structure, classification, and origin identification, as well as enabling the message format to contain more information than such coming from a pure SW integrity protection context.

![Refined generic message format as applicable in NE SW-IP context](image)

In this section we only refer to messages of type SWIP MC, and do not further detail messages of other types. The above message format is slightly different from the format described in [ASMONIA_D22], Section 3.6.1, now including all items, which so far have been mentioned as initial information or as preliminary definition. To give an example, a valid message compliant with the above, final definition could be the following:
Introducing variants of a message (a, b, ..., critical, local, ..., free text) provides an optional means to support further, fine-granular classification, which in many cases may be helpful for pre-sorting messages (one could just order the message variants, but also prioritize them into ‘critical’ or ‘non-critical’ classes). This is an instrument a manufacturer could use for early evaluation of messages (e.g., at NE-Element Manager (NE-EM) level) according to proprietary needs of a specific product, without deeply parsing the message content. To give an example, a variant may express that the message can be handled locally by the NE-EM or must be propagated for further analysis and evaluation.

Note that the above proposal not only introduces a formal syntax, but also aims to provide a very fundamental classification of messages to enable or ease further evaluation. However, the interpretation of events seen in a SW-IP context usually is limited to its direct cryptographic cause, the event time, the validating source and the affected objects. Due to missing context awareness, from method point of view statements on functional repercussions usually are not possible. Nevertheless, the system designer may provide extended information, e.g., when knowing about the importance of an individual SW package and its potential effect on specific features or services on system or network level.

3.3.2 SW-IP for UE

Messages related to the attestation of mobile baseband stacks at network connect time are described in Section 4.3.1.5. To summarize, in this case we need to slightly modify the authentication vectors used in the AKA protocol.

Messages generated on the application processor typically include a hash value of a measured component and can include various additional information. Message are optionally signed and encrypted. In the following an example for a simple measurement report using the ASMONIA IntegrityObservation Scheme is given.

Measurement(Element1, Part1, Context1);
Element(Element1, “application operating system kernel”);
Configuration(Part1, “Linux/Android”, 3.0, valid);
Context(Context1, time, “2013-06-02T21:08:11”);
Context(Context1, imei, “1234567890”);
Context(Context1, hash, “4e1243bd22c66e76c2ba9eddc1f91394e57f9f83”);

3.3.3 Malware Detection for UE

Message formats related to malware detection are using the well known BSON format. Further details are described in Sections 3.1.3.1.2 and 3.1.3.1.4.
4 Integration into Overall ASMONIA Overall Concept

This section describes specific interfaces and particularities, helping to integrate the methods together with their individual SIA into the overall ASMONIA concept.

4.1 Refining the FC-IP Cluster for NE

Abstractly, in the ASMONIA reference architecture [ASMONIA_D11] the functional cluster IP has been introduced as the functionality needed to support collaboration w.r.t. integrity protection for both, UE and NE. In addition several interfaces have been defined, including IPON-Ix, acting as the common (UE, NE), virtual interface between FC-IP and the operator network and IPAC-I acting as the common interface between FC-IP, other functional clusters, and the ACN or ACGW, respectively.

From Figure 1 it is evident that additional, external entities and interfaces are required in order to enable and support FC-IP functions. These have not been considered as part of the ASMONIA reference architecture. Formally completing this we obtain the extended reference architecture as shown in Figure 6:

![Figure 6: Extended Reference Architecture](image)

In accordance with Figure 1 we may map the communication elements from the SIA/NE to the Extended Reference Architecture in the following way:

- Interfaces inside External Entities: 1, 2, 3, 8 (referring to the naming scheme in Figure 1); these interfaces are not further detailed in ASMONIA, but belong to the supporting entities that enable SW integrity protection for NEs.
- IPEXT-I: 4a, 4b, 4c and 7; these interfaces are relevant to deliver SW and HW as well as protected and sensitive data to the MNO and to return faulty NEs back to the factory or to authorized R&S entities.
- IPON-Ix: here: IPON_INE 5a, 5b, 5c and 10b are interfaces inside the operator network used to reliably (and securely) transmit the data required for SW and trust management of protected NEs. They are aligned with the delivery processes, for instance they may be established to transfer SW from readable media (CD, DVD) or from a web-server client to or inside a OAM system.
  6a, 6b and 6c are interfaces mostly internal to the functional elements of the FC-IP
cluster, carrying data, which results from validation processes. Also 9 and 10b may carry such data, but still internal to FC-IP and thus, also assigned to IPON-INE.

- If used, IPAC-I and data streams from and to other functional clusters, via IPMA-I and IPCC-I can be assigned to 10a and also 11 (in other operator domain). From FC-IP point of view IPMA-I is the only interface stringently required for further evaluation of messages, IPCC-I is optional, and IPAC-I is not required.

- The FC-IP (inside the operator network) essentially consists of the assigned interfaces as discussed above and of the SW-IP support functions inside NE-products, the OAM systems, as well as possibly separate operator infrastructure, for instance SW repositories where SW and configuration data may be validated and stored over a long time.

- Interfaces used for collaborative information sharing are not included in the above diagrams, which focus on pure FC-IP aspects. The same applies to MACC_I which is not part of the considerations of this document.

The above view still is very generic and needs to be further detailed, with respect to the 3GPP infrastructure defined between concrete NE and the management systems.

Figure 7: Abstract view on integrating FC-IP / NE functions

An abstraction of the integration of NE software integrity protection functions is shown in Figure 7, where the relation to ASMONIA interfaces to interfaces in NEs and associated OAM systems are shown. This presentation assumes that all FC functions can be integrated via proprietary extensions on top of existing NE functions and the network management infrastructure. Related issues are deepened in Section 4.2, where interfacing of FC-IP functions with a real network infrastructure are taken into account. The graph includes SW-IP aware NEs as well as SW-IP unaware NEs as introduced and explained in [ASMONIA_D22], Section 3.2.2.
4.2 Interfacing of FC-IP functions related to NEs

In Figure 8 a refinement of Figure 7 is shown. Essentially, in the NE case the FC-IP extensions affect the existing NEs as well as their interfacing to OAM systems and also SW-repositories in some cases, where versioned SW is received, stored, and occasionally must be validated during the entire product life cycle. We may assume that no additional traffic in user plane is generated, but all communication is managed by and conveyed over the network management infrastructure.

This part of the NE security infrastructure (for SW-IP) is to a large extend different from the security infrastructure needed for UE (for details and further references see Section 4.3), however, they share the FC-MA as a common component joining and aggregating the message flows for further processing and towards the collaborative ASMONIA information sharing network.

4.2.1 NE towards OAM

Starting from high level, 3GPP defines hierarchy, functionalities and interfaces in several levels for network management, as depicted in Figure 9 (see [TS_32101]). The basic principles also apply to 4G networks. In this figure denote:

- **NE**: Network element representing the managed network element, (and also the primary reporting element as used in the context of this ASMONIA document).
- **EM**: Element Manager, capable to manage a set of NEs of a specific (or compatible) type.
- **DM**: Domain manager, destined to manage a set of EMs within a subnet.
- **NM**: Network Management, “provides a package of end-user functions with the responsibility for the
management of a network, mainly as supported by the EM(s) but it may also involve direct access to the Network Elements. All communication with the network is based on open and well-standardized interfaces supporting management of multi-vendor and multi-technology Network Elements” … [TS_32101].

Likewise, the interfaces between these entities are described as follows:

“A number of management interfaces in a PLMN are identified in figure 1, namely:

1) between the Network Elements (NEs) and the Element Manager (EM) of a single PLMN Organisation;
2) between the Element Manager (EM) and the Network Manager (NM) of a single PLMN Organisation;

NOTE: In certain cases the Element Manager functionality may reside in the NE in which case this interface is directly from NE to Network Manager). These management interfaces are given the reference name Itf-N and are the primary target for standardization.

3) between the Network Managers and the Enterprise Systems of a single PLMN Organisation;
4) between the Network Managers (NMs) of a single PLMN Organisation;
4a) between the Domain Managers (DMs) of a single PLMN Organisation.
5) between Enterprise Systems & Network Managers of different PLMN Organisations;
5a) between the Domain Managers (DMs) of different PLMN Organisations.
6) between Network Elements (NEs).

IRPs\(^3\) may be implemented at interfaces 2 to 5.

The present document identifies Type 1, Type 2 and Type 4 management interfaces. The rest of the 3GPP management specifications focus on Type 2 and to a lesser extent on Type 1 management interfaces. In addition, the rest of the 3GPP management specifications will not refer to Type 4 management interfaces.

---

\(^3\) The acronym IRP denotes: Integration Reference Point
interface. Specific Type 2 protocols and information model that are applicable for use in Type 4 management interface are listed in Annex E.

The present document identifies Types 3, 5 & 5a management interfaces. Detailed specification of these interfaces is For Further Study (FFS).

The specification of the management interfaces of type 4 & 6 is beyond the scope of standardisation.”

Due to the conditions given by the status of TS-32-series standardization, in this document we will restrict on type 1 and type 2 interfaces (which are the major targets for standardization) and we assume that on higher levels data and associated control flows can be arranged in a way that requirements and needs for type 1 and 2 interfaces and thus, the managed objects, can be met. In the context of the FC-MA (as shown in Figure 8 and Figure 9) 4 and 4a interfacing would be interesting to consider as well, but is not well detailed yet in related standardization. However, as it is stated

The approach for Interfaces of type 4a (the Itf-P2P interface) is the same as for interfaces of type 2 (the Itf-N interface – see clause 5.1.2.2).

The Itf-P2P should as much as possible re-use the interface definitions of the Itf-N interface.

Further details on the Itf-P2P interface are available in 3GPP TR 32.806 [107].

focusing on type 1 and 2 actually seems to be sufficient. In any case it should be possible to fulfill ASMONIA requirements, e.g., on collecting messages from NEs, via proprietary extensions, implemented in NM or even particular DM entities. In [TS_32101], Annex A, the following protocols are mentioned at application layer (type 1), applicable between NE and EM and at network layer (Annex B):

TS 32 101 Annex B for network-layer protocols:
The valid Management-application-layer-protocols for 3GPP are:
- CORBA IIOP (see references [8] and [52]);
- NETCONF (see reference [118]);
- SNMP (see reference [6]);
- SOAP (see references [108] and [109]).

The valid Management-application-layer-protocols for bulk & file transfer are:
- FTAM (see references [13] – [19]);
- ftp (see reference [4]);
- tftp (see reference [5]);
- sftp (secure ftp).

The valid Management-application-layer-protocol for Home NodeB Management Interface Type 1 and Home eNodeB Management Interface Type 1 is:
- TR-069 (see reference [115])

The valid Management-application-layer-protocols for bulk & file transfer for Home NodeB Management Interface Type 1 and Home eNodeB Management Interface Type 1 are defined in TR-069 [115].

TS 32 101 Annex B for network-layer protocols:
The valid network layer protocols for the management of 3GPP are:
- IP (see reference [48]);
- X.25 (see reference [22]).

NOTE 1: IP is the recommended networking protocol.

NOTE 2: Normative references relating to ISO Transport over TCP/IP are [46] and [47] and ISO Transport over X.25 are [43] [45].
Below we examine typical examples for such layer 1 interfaces, to assess their applicability and extensibility in the ASMONIA context.

4.2.1.1 SNMP

The Simple Network Management Protocol SNMP is defined by [RFC_1157]. SNMP is widely used for monitoring and management of network-attached devices (in the context of this paper this conforms to NEs) from a central management console (conforms to EM). The generic architecture is shown in Figure 10, were the essential entities and protocol elements are visible. Typically a managed device (NE) consists of several SW agents or HW cooperating with a master agent, which is responsible for the communication with the manager (EM). The agents are specific, proprietary tasks to monitor or control the system they are running on. The internal communication between Master Agent and Agents is not specified and can be designed as needed for a specific purpose.

![SNMP Architecture Diagram](http://en.wikipedia.org/wiki/Files:Snmp.PNG)

**Figure 10: Generic SNMP Architecture**

SNMP defines flexible protocol elements (GET/SET requests, GET/SET responses, TRAPs) usually sent over UDP, but it does not specify the conveyed content. GET elements are used to call for a management data set and SET elements are used to send data, which controls settings of the managed system. A TRAP is a data set sent by the managed device in case of unexpected or urgent events that have to be reported to the management system. While usually initiative for the communication is driven from the manager (resulting in GET/SET request – GET/SET responses), TRAPS allow unasked notification initiated by the master agent and thus enable the manager to react with GET/SET activities if needed or with other appropriate behavior (alerts, alarm display, etc.).

As mentioned, the content (structure and semantic) of SNMP packets is not specified through [RFC_1157], but connected via a so-called Management Information Base (MIB). While several predefined MIBs exists (such as, e.g., MIB2\(^5\)), each manufacturer is free to define proprietary MIBs or extensions to existing ones, so that any data required to control or to monitor a managed device can be captured. If compatibility with other network elements is required such extensions can become centrally registered at IANA\(^6\) assigning new object identifiers (OID), i.e., data elements, within the tree-like MIB structure. Once an OID is officially registered, its meaning and content must not be changed anymore. Of course, private

\(^6\) [http://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority](http://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority)
MIPs may exist and could be applied, as long as their usage does not cause incompatibilities with non-proprietary or standardized equipment.

Taking these capabilities of SNMP it is easily possible to specify agents and MIB elements in a way that they are able to convey and handle ASMONIA specific content such as introduced in [ASMONIA_D22], Section 3.6.

Even messages for dedicated protocols (round-trips) can be included, such as mentioned in [ASMONIA_D22], Section 3.7.2.4 and 3.7.2.7, by matching GET/SET requests and responses. The fact that those messages contain signed data virtually forming kind of a session does not harm, as long as the sequence of sent data is respected, concatenating individual ‘request – response’ pairs in a defined order. This is a matter of the internal logic of the management components and not restricted by SNMP.

It may be a subject of further research, whether security extensions for SNMP, such as defined with [RFC_1352] can be applied. However, this is more seen to be a subject for optimizing implementation with help of predefined work, than a fact causing any restrictions to the SW-IP methods considered in ASMONIA.

As SNMP is widely used in OAM context this holds true for a large number of NEs.

### 4.2.1.2 Bulk & File Transfer (SW and Trust Management and Updates)

Usually file transfer protocols are taken to enable large SW or data uploads, as indicated by [TS_32011], Annex A. For usage of a file transfer protocol, such as FTP or SFTP, it is not relevant if a file is signed and packed (zipped or archived) together with an accompanying signed object and certificates. Thus, we can assume that integrity protected SW uploads will be supported by each of these bulk and file transfer mechanisms, independent from the semantic meaning of the file content itself (SW, FW, configuration data, etc.). As signed objects protect an associated file independent from the underlying media it may not even necessary to apply any transmission security, unless this is required for other purposes, such as confidentiality (e.g., SW containing secret keys) or non-repudiation. uch security requirements can be fulfilled independent from the transfer protocol, for instance by sending an encrypted .zip file. Thus, we can state, that SW management as well as trust management in cases, where this is done based on files (such as scripts, configuration data, SW packages, etc.) easily can be accomplished via type 1 interfacing. Non repudiation may require bi-lateral transmission of data (e.g., sending a confirmation back to an initiator of security transaction), however, this may also be done using SNMP in parallel. To give an example, a SW package containing a CRL could be securely announced and confirmed via SNMP, while the bulk data itself is send via FTP; thus the initiator will know that the sensitive data was received. Such issues are matter of the internal logic between a specific NE and NE manager.

#### 4.2.1.3 TR-069

We put specific attention to [TR_069], as this is the standardized generic management protocol applied (if needed, over a specified TLS profile) between H(e)NB and H(e)MS. It is mandatorily required for HnB management and particularly (in 4G context) for HeNBs, as stated in [TS_33320], Section 8.4.3:

For the management of the H(e)NB by the H(e)MS, the CPE WAN Management Protocol TR-069 [15] shall be used with the following restrictions and extensions:

The TLS profile specified in TS 33.310 [7], Annex E, shall apply.

- **Shared-secret-based authentication between H(e)NB acting as CPE and H(e)MS acting as ACS shall not be allowed.** Only certificate-based authentication shall be allowed.

- The use of TLS to transport the CPE WAN Management Protocol shall be mandatory in case that the H(e)MS is accessible on public internet or when TLS is used within the IPsec tunnel.
Further, according to [TS_33320], Section 8.4., for SW downloads the following applies:

The H(e)NB shall utilize the established TR-069 method to download software from the H(e)MS or a server directed to by the H(e)MS according to TR-069 Version 1 Amendment 2 [15]. The following requirements are added for security:

- The file shall use the signed package format according to TR-069 [15].

NOTE 1: Depending on the link to H(e)MS, transport security is provided by the secure link according to clauses 4.3.1 (when H(e)MS is in operator network) or 4.3.2 (when H(e)MS is in public Internet).

- The SignedData object in the signed package shall contain at least one signature provided by a software signing entity, with certificate issued by an operator trusted CA.

- The TrE shall use a public key issued by an operator trusted CA to verify the software signing entity certificate and the signature(s) in the SignedData object. If the verification fails, the H(e)NB shall delete and not install the software that failed the verification. All root certificates used for this purpose shall be stored in the TrE.

NOTE 2: Notification of this failure is according to procedures in TR-069 [15].

- The signed package shall also contain the trusted reference values needed for the software integrity checks performed during secure boot.

NOTE 3: TR-069 [15] supports multiple signatures which can be used for the purpose of supporting different hash algorithms.

TR-069 based integrity measures are in accordance with the SWIP methods as proposed in ASMONIA in NE context, as these rely on a PKI backbone. Note that for native TPM based integrity protection at this point additional security measures (signatures) and infrastructure (management of asymmetric cryptography) had to be applied, which are not part of TCG concepts!

TR-069 specifies details on the signed package format in [TR-069], Annex E, based on the data structure as shown in Figure 11:

![Figure 11: TR-069 Signed Package Format](image)

The Signed Package Format also implies a command list, which may consist of either attributes (e.g., SW version) or actions to be taken (e.g., update, move, remove, reboot, ..) after download. The command list can be extended by vendor specific commands, thus it is possible to apply a generic command structure.

Signatures (which, according to TS_33320, must be one or more) are required in PKCS#7 format, which also support signatures based on X.509 certificates that can be attached.

It is further stated:

---

7 PKCS7 is described in RFC 2315; meanwhile 2315 is substituted by RFC 5652 (CMS); [http://tools.ietf.org/html/rfc5652](http://tools.ietf.org/html/rfc5652), but in context of TR-069 it still applies.
The signatures are “external signatures,” meaning that the signed message is not encapsulated within the Signed Data object. Instead, the signed message data consists of the octet string formed by the header and the command list components of the package.

At first view this sounds a bit strange as the payload itself seems not to be included in the signature. However, looking into the command list reveals that hashes of the payload are ‘hidden’ as parameters of individual commands associated to a specific payload. This seems to be a formal restriction imposing signature generation in TR-069 specific way, even if Signed Package Formats could be factory-prepared before massed downloads or could be created during individual download. Nevertheless, it would also be possible to define generic (vendor specific) commands saying “use verification and enforcement mechanisms specified by the referenced payload itself” and then apply these in addition to the verification of a Signed Package data structure (referencing the payload in such command), which would be created at download time (e.g., based on an operator PKI). The payload could contain another vendor specific signed objects and related certificates (independent from TR-069, based on a vendor PKI), e.g., if an XML based signature format is preferred to be used by the verification logic. This way, a managed device would only accept SW downloaded from a specific OAM server, which could select between several signed SW versions provided by the vendor. As specific commands can also be defined for trust management (e.g., on trust anchor management), TR-069 seems not to imply restrictions for the applied end-to-end cryptography.

However, it is still unclear, whether and how it is possible to send messages back to the H(e)MS, e.g., if this is required for any security reason (either to report integrity breaches at runtime or after successful boot or in case of hand-shaked messages that have to be sent establish a two way communication context).

One of the initial TR-069 use cases is status and performance monitoring. In [TR_069], Section 1.3.3. it reads:

The CPE WAN Management Protocol provides support for a CPE to make available information that the ACS may use to monitor the CPE’s status and performance statistics. It also defines a set of mechanisms that allow the CPE to actively notify the ACS of changes to its state.

Apparently, this is what we need to fulfill the ASMONIA demand on reporting an NE’s integrity state. TR-069 also knows “sessions,” which can be initiated by the CPE (Customer Premises Equipment, i.e., the H(e)NB), see [TR_069], Section 2.3.3 and 3.2.1, either triggered by events or in a periodic fashion. This can be accomplished via Remote Procedure Calls (RPC). It is also possible to initiate a communication from ACS (Auto Configuration Server, i.e., the H(e)MS) side, by sending connection requests to the CPE, see Section 2.3.4. and 3.2.2. Moreover, the CPE MUST initiate connections to the ACS in well defined situations, e.g. after on power up or reset, on initial installation, on ACS Connection Request. Thus, the basic requirements are met.

The next question now to be answered is ‘which type of data can be sent’ as the data model associated with TR-069 knows many pre-defined data sets. It has to be clarified whether messages, as needed for ASMONIA could be launched. Specific information can be found in TR-069 itself as well as in TR-106 [TR_106], which defines a generic data model.

---

8 usually neither header nor command parameter contains individual attributes, such as time stamps or target IDs
9 Compare TR-069, session: A contiguous sequence of CWMP transactions between a CPE and an ACS. Note that a Session may span multiple TCP connections.
Looking into Figure 12 shows us the protocol stack and explains a typical TR-069 based communication relying on a secure TLS channel. SOAP is used as the formal container to convey RPC calls/answers and parameters, which are transmitted via HTTP posts and responses into the requested direction. Note that usually the CPE (H(e)NB) initiated such communication but the same sequence could be enforced by the ACS (H(e)MS) be sending a preceding Connection Request. Also note that the initiative of invoking an RPC method can be reversed and given back to the communication partner, e.g., when sending empty HTTP requests and responses. An empty HTTP message basically means that there are no more RPC requests pending from partner A (e.g., the CPE) and if the connection is not closed, it offers the opportunity to launch an RPC by partner B (e.g., the ACS).

One possibility to send SW-IP-related messages is offered by TR-106 using the Common Object Definition for the Device.DeviceInfo.DeviceLog elements, as listed in Table 3 in [TR_106] on page 22-24, and illustrated in Figure 13. This element allows vendor-specific log strings, which could encapsulate ASMONIA messages, sent from the HeNB to the HeMS.
Establishing UE and NE Protection Methods  
Security Infrastructure Integration and Re-Evaluation  
D23-1.0

DeviceDeviceInfo.DeviceLog elements could, e.g., be sent in InformRequests (which MUST be sent before each session) by the HeNB, where the managed device first informs about the reason or event to trigger this session.

Another possibility would be to apply vendor-specific methods (RPC functions) supporting SW-IP type of messages, even if this requires extensions for both, CPE/HeNB as well as ACS/HeMS. TR-069 (in version 1.4) explicitly states:

1.4 Architectural Goals
---
The protocol should allow vendor-specific parameters to be defined and accessed
---
The protocol also includes an extensibility mechanism that allows use of vendor-specific Parameters in addition to those defined in this specification
---
The protocol is also designed to be extensible. It includes mechanisms to support future extensions to the standard, as well as explicit mechanisms for vendor-specific extensions.
---

A.3.1.1 GetRPCMethods
This method MAY be used by a CPE or ACS to discover the set of methods supported by the ACS or CPE it is in communication with. This list MUST include all the supported methods, both standard methods (those defined in this specification or a subsequent version) and vendor-specific methods. The receiver of the response MUST ignore any unrecognized methods.

Similary, in TR 106 it is stated:

TR 106:
3.3 Vendor-Specific Parameters
A vendor MAY extend the standardized parameter list with vendor-specific parameters and objects. Vendor-specific parameters and objects MAY be defined either in a separate naming hierarchy or within the standardized naming hierarchy.
---
The full path name of a vendor-specific parameter or object MUST NOT exceed 256 characters in length.
---
When appropriate, a vendor MAY also extend the set of values of an enumeration. If this is done, the

<table>
<thead>
<tr>
<th>DeviceDeviceInfo</th>
<th>object</th>
<th>-</th>
<th>This object contains general device information.</th>
<th>-</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>string</td>
<td>-</td>
<td>The manufacturer of the CPE (human readable string)</td>
<td>-</td>
<td>1.0</td>
</tr>
<tr>
<td>ManufacturerCDI</td>
<td>string</td>
<td>-</td>
<td>Organizational unique identifier of the device (manufacturer. Represented as a six hexadecimal digit value using all upper-case letters and including any</td>
<td>-</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DeviceStatus</th>
<th>string</th>
<th>-</th>
<th>Current operational status of the device. Enumeration of UP</th>
<th>-</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>IsAvailable</td>
<td>-</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>-</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Disabled</td>
<td>-</td>
<td>1.0</td>
</tr>
</tbody>
</table>

| UpTime           | unsignedInt | - | Time in seconds since the CPE was last restarted | - | 1.0 |

| FirstUseDate     | date/time | - | Date and time in UTC that the CPE first successfully established an IP-layer network connection and acquired an absolute time reference using NTP or equivalent over that network connection. The CPE MAY use this date after a factory reset. If NTP or equivalent is not available, this parameter, if present, SHOULD be set to the unknown Time value. | - | 1.0 |

| DeviceLog        | string[32/16] | - | Vendor-specific log(id) | - | 1.0 |

Figure 13: TR 106 Objects

Copyright © 2013 ASMONIA consortium. All rights reserved.
vendor-specified values MUST be in the form "X_<VENDOR>_VendorSpecificValue". The total length of such a string MUST NOT exceed 31 characters.

Summarizing the above, using TR-069 and associated standards there are no blocking-points seen, neither for trust or SW management nor for collecting messages coming from HeNBs in the HeMS. Thus HeMS could serve as aggregation points for HeNBs as needed in ASMONIA context.

There may be more ways of encapsulating SW-IP notifications in other TR-106 data elements, such as for fault management, but this has not been examined in detail and thus, is left an issue for specific implementations.

4.2.2 OAM towards FC-MA

Above it has been described how information and data required to control SWIP methods in NE and to collect messages can be conveyed via type 1 interfaces. The underlying problem was to verify whether applied protocols allow integration of SWIP-related messages and protocols between NE and NE-Element manager (NE-EM) in the OAM subsystem. Taking SNMP and TR069 as relevant examples, it has been shown that the integration is possible and even multi-message protocols (virtual sessions) could be handled at this level. The NE-EM may need implementation of proprietary extensions, in order to execute low level control and monitoring. This is specific to an individual NE element and SWIP use case and will not be described in further detail (these functions follow the SWIP governance principles as described earlier). Proprietary OAM extensions in NE and NE-EM are not seen as a problem, as already today this is common practice.

Given the NE-ME handles all the operations required for trust management based on information obtained from a superior OAM-Logic (which can be extern to the mobile NW and e.g., be driven from the manufacturer’s or operator’s business logic), between NE-EM and upper layers (DM, NM, ..), there is no need for specific extensions except for directing the dataflow and for storing of information. But our assumption is that this can be managed applying existing OAM-functionalities and without SW-IP related adaptations. Based on these ideas, Figure 14 shows the resulting dataflow required to integrate SW-IP into a mobile network.

![Diagram showing dataflow for SW-IP integration](image)

*Figure 14: Dataflow to monitor and control SW-IP for NEs*

At the right the communication with FC-MA is illustrated, which essentially is realized by aggregating and converting SW-IP messages towards FC-MA. ‘New’ in this context is the conversion function CONV, which decouples proprietary semantic and syntax of the messages generated by (vendor specific) NE and/or NE-EM elements by converting these into an ASMONIA wide data format. To enable this, the CONV function is feed be NE-specific format descriptors containing a complete rule set for message-driven conversion processes. In most...
Establishing UE and NE Protection Methods
Security Infrastructure Integration and Re-Evaluation
D23-1.0

cases such a descriptor may get implemented as sort of vendor-provided script (e.g., python, bash, XML-based) or compiled or interpreted program (Java, C,..) that can be executed by the CONV entity without further implementation efforts. That way NE manufacturers can decide on own extensions, protocols and message formats at lower level, but at the same time they can integrate with the ASMONIA FC-MA function without the need for standardization at message level. Nevertheless, some platform related arrangements are necessary, e.g., on the execution framework of the CONV service.

However, this could be managed by agreements between operators and manufacturers and need not to become matter of standardization. To given an example, in the simplest case a server, capable to run python scripts and knowing how to communicate with the CONV databases would be sufficient. Based on such an agreement, format conversion scripts could easily be generated for each SW-IP use case associated with a manufacturer specific NE.

4.3 Interfacing of FC-IP functions related to UEs

Here we focus on measures for SW integrity protection for UEs and debate how these can be integrated into a mobile network and into the overall ASMONIA concept. As already mentioned earlier (Section 3.1.2) no infrastructure for UE management already exists for such purpose – thus integration appears to be more specific as compared to the re-use of the OAM infrastructure for NEs. SW-IP for UEs requires additional functions and entities, similar to the integration of malware detection methods. In the figure below architectural principles are shown, where light-blue (for baseband: CT-SW-IP collector, Interface + control, connect-time SW-IP extensions in UE) and green components (for applications: runtime SW-IP-extensions in UE, RT SW-IP collector) as well as common components (RA\textsuperscript{10} Service, database for trusted values) are assigned to SW-IP and magenta components (MW extensions in UE, MW controller/collector) to MWD functionality. Details are explained in the following subsections.

\textbf{Figure 15: Overview on UE based FC-IP functions}

\textsuperscript{10} RA stands for ‘remote attestation’ as known from TCG context
4.3.1 Backend Functions associated with Integrity Protection for UEs

In this section, we describe the attestation of mobile baseband stacks in more detail, as first presented in [WWS2012].

To prevent attacks on networks based on compromised or non-specification-compliant baseband stacks, we propose an attestation protocol to verify the trustworthiness of a mobile device before it can communicate with the core components of a mobile network, such as the HSS.

The main idea is that only trustworthy mobile devices are allowed to fully access the critical components of a network. To demonstrate its trustworthiness, the baseband stack running on a mobile phone has to prove its authenticity and integrity. If the attestation procedure fails, the network only allows limited access, e.g., download a trustworthy version of the baseband stack, which is signed by its vendor, to replace the compromised one. That way, the proposed attestation protocol allows the mobile device to recover from malicious modifications and protects the network from attacks by compromised mobile devices.

4.3.1.1 Notation

A cryptographic hash function \( H \) is a one-way function with collision and pre-image resistance that compresses input data of virtually arbitrary length to a fixed sized output of length \( l \), that is \( H: \{0,1\}^* \rightarrow \{0,1\}^l \). Applying the hash function \( H \) to data \( m \) is denoted as \( H(m) \), which generates a hash \( h \).

A message authentication code MAC is a function that calculates a message digest with fixed length \( l \) for an input \( m \) with virtually arbitrary size based on a secret key \( K \): \( MAC(K,m) = \text{dig} \). The resulting digest provides information that can be used to verify both the integrity as well as the authenticity of the message. An HMACa specific way to construct a MAC based on a hash function, i.e., \( HMAC(K,m) = H((K \oplus \text{opad}) || H((K \oplus \text{ipad}) || m)) \), where \( || \) denotes concatenation, \( \oplus \) the exclusive or, \( \text{opad} \) the outer and \( \text{ipad} \) the inner pad.

A particular system state is represented by a set of integrity measurements stored in the (currently up to 24) hardware-protected PCR of an MTM. Such a set of PCR describing a system state is referred to as platform configuration \( P := (PCR[0], \ldots, PCR[24]) \). After a system reset, the contents of all PCR is set to zero, i.e., \( PCR[i] \leftarrow 0 \ \forall i < 24 \). To store a fresh integrity measurement \( \mu \) in a PCR with index \( i \), the current value is combined with the fresh measurement value using \( PCRExtract(PCR[i], \mu) \), which is specified as \( PCR[i] \leftarrow SHA1(PCR[i] || \mu) \) [TCG_TPM].

Wrapping a cryptographic key \( K \) with a public asymmetric key \( pk \) to a specific platform configuration \( P \) is denoted as \( \{K\}^P.pk \) and essentially encrypts the key. To decrypt the wrapped key with the corresponding private key \( sk \), the actual configuration \( P' \) needs to match exactly the specified configuration \( P \).

To encrypt and bind arbitrary data \( m \) to a platform configuration \( P \), the MTM essentially provides a TPM_Sea l command, which is referred to as Seal for the sake of simplicity. With Unseal, which is short for TPM_Unseal, the MTM/TPM can decrypt the data \( m \) if the system is in a state which matches the specified platform configuration \( P \). Given a non-migratable asymmetric key \( K_{seal} = (pk, sks) \), we denote the result of sealing data \( m \) to the platform configuration \( P \) with \( \{m\}^P.pk = Seal(P, pk, sk, m) \). To unseal the sealed data \( \{m\}^P.pk \), it is required that the actual platform configuration \( P' \) is equal to the specified platform configuration \( P \): \( m = Unseal(P' = P, sks, \{m\}^P.pk) \).
4.3.1.2 Cryptographic Keys

We define a non-migratable asymmetric wrapping key $K_{\text{wrap}} = (pk, sk)$ and sealing key $K_{\text{seal}} = (pks, sks)$, where $pk$ and $pks$ are the public keys while $sk$ and $sks$ are the secret keys of the respective keys. Both asymmetric keys are securely generated and stored inside the MTM.

With the public key $pks$, a public signing key $pk_{\text{SigBV}}$ sealed to a platform configuration $P_L$, where all PCR are selected, but have the value zero, except for $\text{PCR}[0]$, which stores the measurement value of the boot loader $L$. As a result, the sealed key $\{pk_{\text{SigBV}}\}_{P_L,pks}$ only be unsealed by the boot loader, which is stored in ROM and executed first. This unsealed key $pk_{\text{SigBV}}$ used in case of an update to verify a signature $\text{sig}$ of the baseband update before installation.

With the public key $pk$, an asymmetric integrity key $K_{\text{INT}}$ wrapped to a trusted platform configuration $P_B$ \{($K_{\text{INT}}$)$_{P_B,pk}$, where $B$ denotes the baseband. This wrapped key is used in our protocol to verify the integrity of the baseband. It is important to note that the platform configuration $P_B$ $P_L$ no longer be used as a valid platform configuration for cryptographic operations, e.g., unseal a key.

Together with the authentication data $A_{\text{seal}}$, the sealing key $K_{\text{seal}}$, both keys are stored in flash memory in encrypted form (sealed or wrapped) during initialization. However, having $A_{\text{seal}}$ flash is not a security problem, because the key $K_{\text{seal}}$ protected by $P_L$, thus it is only available to the boot loader $L$.

The USIM holds the key $K_{\text{an attestation key}} K_{\text{ATT}}$, which are both shared with the AuC, as well as the passphrase $A_{\text{wrap}}$ wrapping key $K_{\text{wrap}}$.

4.3.1.3 Concept and Main Ideas

In contrast to existing hardware-based attestation protocols, which are often based on asymmetric cryptographic operations provided by an MTM [Yingyou2010], we propose a symmetric approach to efficiently prove the trustworthiness of a mobile device, in particular, its baseband stack.

As most existing protocols, we rely on an authenticated boot process starting from a CRTM: The current software binary calculates a hash of the following binary in the boot chain and stores it as an integrity measurement value in a platform configuration register of the TPM or MTM before executing the measured binary.

In our concept, the boot loader acts as CRTM for the sake of simplicity, which is why it must be stored in ROM as indicated in Figure 16 (top left), which shows our system architecture. Together with the baseband binary (top right), Figure 16 also depicts a boot procedure in the top and the relevant hardware components, namely the flash memory (bottom left), the MTM (bottom center), the USIM (bottom right), and the baseband processor with RAM, in the bottom half.
Establishing UE and NE Protection Methods  
Security Infrastructure Integration and Re-Evaluation  
D23-1.0

Figure 16: System Architecture [WWS2012]

In the boot procedure, the loader \( L \) first measures itself and extends the \( PCR[boot \, loader] \) (steps 1 and 2) creating the platform configuration \( P_L \). In case of an update, the loader unseals the public signing key \( pkSigBV \), verifies signature \( sig \) of the new baseband binary with the unsealed \( pkSigBV \) (step 3) and re-wraps the integrity key \( K^{INT} \). In the process, the unseal operation implicitly validates the integrity of the boot loader \( L \) (represented by \( P_L \)) and the public signing key verifies the new baseband with its signature. After that, the boot loader measures the baseband (step 4) and extends the \( PCR[baseband] \) (step 5), which creates the trusted platform configuration \( P_B \). Finally, it executes the baseband binary (step 6).

For a remote attestation, the TPM or MTM normally signs the list of PCR representing the current platform configuration and sends it to a remote verifier. Based on the values (and a so-called measurement log), the remote party can then decide whether the platform is still trustworthy. We call this mechanism explicit attestation, because the complete measurement log needs to be transferred.

For our protocol, however, we adapt the concepts of implicit attestation, which does not need to transfer the measurement log. Instead, implicit attestation usually relies on some pre-shared (authentication) information, such as a sealed symmetric key or hash chain [Krauss2007], which can only be accessed, if the platform is still trustworthy. This approach is, for instance, used to validate whether the boot loader is in a trusted state before a new baseband binary is loaded in case of an update (step 3). So, as long as the prover can successfully authenticate itself, the verifier has implicit proof of the integrity of the prover’s system.

However, most existing implicit attestation protocols still rely on relatively expensive asymmetric cryptographic operations, such as signing or unsealing a sealed key. That is why we propose a more efficient approach based on symmetric cryptographic operations to implicitly prove the trustworthiness of a mobile device, especially its baseband stack.

The main idea is that the USIM only grants access to the attestation key \( K^{ATT} \) which is necessary to calculate an attestation response for the network if the baseband stack is trustworthy. To prove its trustworthiness, the baseband merely needs to load the integrity key \( K^{INT} \).
which is cryptographically bound (wrapped) to a trusted platform configuration $P_B$ based on a signed baseband stack. Note that the initial wrap operation is necessary only once during initialization and thus has no direct influence on the overall efficiency of our attestation protocol. More important, the MTM only needs to actually unwrap the wrapped key if the key is not yet decrypted and loaded, because the key itself is never used for security critical operations. Finally, to be able to securely verify the baseband stack, we moved the calculation of authentication value, which is an HMAC based on $A_{\text{wrap}}$, needed to load the key $K_{\text{INT}}$, inside the USIM. So, the baseband has to request the correct authentication value from the USIM before it can load the key inside the MTM. If the load operation was successful, the USIM can verify the HMAC-authenticated result, which is protected by $A_{\text{wrap}}$.

4.3.1.4 Integrity Verification of the Baseband Stack

For a local attestation (integrity verification) between the baseband stack (prover) and the USIM (verifier), which is depicted in detail in Figure 17, the baseband system (center) loads the wrapped key $\{K_{\text{INT}}\}^P_B$ the MTM (right) and the USIM (left) verifies the HMAC-authenticated result. However, since we moved the HMAC calculation for the authentication and verification (steps 3 and 10) inside the USIM, all security critical operations are performed inside one of the hardware secure elements.

![Figure 17: Attestation of Baseband towards USIM [WWS2012]](image)

Usually, a dedicated trusted software stack is responsible to calculate and assemble the necessary parts of the command, such as the authentication value $\text{parentAuth} (pA)$ for the wrapping key $K_{\text{wrap}}$ (Figure 17, step 3), and send the complete command structure to the MTM. In our protocol, however, the main idea is to move this calculation of the authentication value ($pA$, steps 2 and 3), which authorizes the use of the wrapping key $K_{\text{wrap}}$, inside the USIM. That way, the passphrase $A_{\text{wrap}}$ leaves the secure environment of the USIM firmware.

As a result, the USIM needs to securely generate the required authentication value $pA$ on behalf of the usual software component, which can be implemented in software and easily integrated into the existing USIM firmware. The value for $\text{parentAuth}$ is generated from concatenated HMAC inputs (denoted by $1H1$ to $4H1$) as
\[ pA = \text{HMAC}(A_{\text{wrap}}1H1 \parallel 2H1 \parallel 3H1 \parallel 4H1), \] (1)

where

\[ 1H1 = H(\text{TPM\_ORD\_LoadKey2} \parallel \{K^{\text{INT}}\}_{\text{P\_B, pk}}) \]
\[ 2H1 = \text{authLastNonceEven} \]
\[ 3H1 = \text{nonceOdd, and} \]
\[ 4H1 = \text{continueAuthSession}, \]

according to the TCG specification [TCG\_TPM].

When the MTM receives the load command, it internally verifies the authentication value \( pA \) and matches the specified platform configuration \( P_B \) against the actual platform configuration \( P \) (step 5). If the equation \( P_B = P' \) holds, the MTM loads the key. For efficiency reasons, the MTM should only verify the pre-conditions, e.g., platform configuration and authentication data, and not actually decrypt the key if the key is already loaded. The MTM then calculates a result message, which includes a specified return code, e.g., \( \text{TPM\_SUCCESS} \), the \( \text{nonceOdd} \), and a second HMAC \( \text{resAuth} \) to authenticate the response (step 8).

To complete the attestation procedure, the USIM receives the result of the load operation \( \text{TPM\_LoadKey2} \{K^{\text{INT}}\}_{\text{P\_B, pk}}, pA \) and merely needs to verify the return message: For that purpose, the USIM compares the output \( \text{nonceOdd} \) with the input \( \text{nonceOdd} \), which must be exactly the same and prevents replay attacks (step 9). By recalculating and checking the HMAC \( \text{resAuth} \) (and the return code), the USIM can efficiently verify whether the key was correctly loaded, thus, stating that \( P_B = P' \) (step 10). The fresh HMAC \( \text{resAuth}' \) is calculated again according to Equation 1, where

\[ 1H1 = H(\text{returnCode} \parallel \text{TPM\_ORD\_LoadKey2}) \]
\[ 2H1 = \text{nonceEven} \]
\[ 3H1 = \text{nonceOdd, and} \]
\[ 4H1 = \text{continueAuthSession} \]

as specified by the TCG [TCG\_TPM]. If the load operation has been successful, which is indicated by the \( \text{returnCode} \), the verifier has implicitly proven that the baseband stack is still unmodified and has not been compromised. As shown in Figure 17, the USIM now allows access to the attestation key \( K^{\text{ATT}} \), which is limited to the current AKA protocol run indicated by the random number \( \text{RAND} \).

4.3.1.5 Generation of authentication Vectors

Based on the result of a local baseband attestation, the USIM (now prover) is able to provide proof of the baseband’s trustworthiness towards the network (verifier). We only need to slightly modify the AV used in the AKA protocol. Depending on the network type (3G or 4G), the AV are usually generated as

\[ \text{UMTS\_AV} := (\text{RAND} \parallel \text{XRES} \parallel \text{CK} \parallel \text{IK} \parallel \text{AUTN}) \] or \( \text{EPS\_AV} := (\text{RAND} \parallel \text{XRES} \parallel K^{\text{ASME}} \parallel \text{AUTN}), \] (2) (3)

where \( \text{RAND} \) is a random number, \( \text{XRES} \) is the pre-calculated (expected) authentication result, \( \text{CK} \) is a confidentiality and \( \text{IK} \) an integrity key, and \( \text{AUTN} \) an authentication token. These components are calculated as depicted in Figure 18, where \( f1 \) and \( f2 \) are MAC and \( f3 \) to \( f5 \) as well as KDF are key derivation functions.
In our concept, we add an expected attestation value ($X_{ATT}$) to the AV, which allows the Service Network to verify the trustworthiness of the mobile device. This additional attestation value is generated from the random number $RAND$ with a dedicated attestation key $K_{ATT}$ (which is only available, if the local baseband attestation has been successful), that is

$$X_{ATT} = HMAC(K_{ATT}, RAND) .$$

(4)

Since the symmetric attestation key $K_{ATT}$ only known to the HSS (and the USIM, of course), the Home Network has to pre-calculate the HMAC for the Service Network. That way, the SN can compare $ATT$ (from the USIM) with the expected attestation value $X_{ATT}$ without the knowledge of $K_{ATT}$.

We also send a second attestation value $ATT_B$ from the mobile device to the Home Network, which is generated based on the hash value of the baseband stack $h_B$ some information $BI$, for instance, about the version, state, and configuration of the baseband. These values can be used by the home network to further evaluate the baseband stack and enforce a certain version or configuration.
Figure 19: AKA-based Attestation of Baseband / USIM towards the Network (simplified) [WSS2012]

As shown in Figure 19, we define the following attestation-based access policy: If the response \( RES = f_{K_A}(RAND) \) from the USIM matches the expected response \( XRES \) and the attestation value \( ATT = HMAC(K_{ATT}, RAND) \) also corresponds with the expected value \( XATT \), the mobile device can fully access the network (steps 8–9). However, if the (local) attestation fails, the network only grants limited access, e.g., download a signed recovery version to replace the modified baseband stack. By sending the hash of the baseband information \( Bl \), which are protected by the second attestation value \( ATT_B \) (step 7), the Home Network can evaluate the configuration of the baseband in detail (steps 10–11). As a consequence, particular services or operations involving critical network components could be allowed (or denied). The Home Network could even enforce a certain baseband version by simply evaluating the baseband version in \( Bl \) and restricting access for unsupported versions.

4.3.2 Backend Functions associated with Anomaly Detection (Malware) for UEs

Based on the closer UE context for malware detection in Section 3.1.3 in the following we describe network level mechanisms to support the UE-centric anomaly detection as proposed in [ASMONIA_D22]. This section focuses on a novel backend technology based on Malware Controller/Collectors and related strategies that are required to integrate MWD mechanisms inside an MNO domain. Other malware related in-network mechanisms are also described in [ASMONIA_D42ii].

4.3.2.1 Communication and Interfaces towards MN

This section details the communication patterns between the UE based anomaly detection and its MNO based support mechanisms. Figure 21 shows the overall setup and the involved components. For simplicity we assume that the communication between UE and the RMWCC is based on the Internet Protocol (IP).
The communication between the UE and the RMWCC could either be designed using the available standardized interfaces of the current standard, or by introducing new interfaces for this specific purpose.

In this document we only consider the interfaces and entities in a standard 4G setting. Accordingly, the preferred interfaces and entities are depicted in Figure 20.

As the UE is able to communicate to the Serving Gateway (SGW) via an IP connection, we believe that it would be beneficial to rely on this essential functionality. The regional SGW can then route the anomaly detection event traffic to the responsible RMWCC. GTP-U v1 running on UDP may be used as the application protocol that carries the UE payload.

Figure 20: Entities and Interfaces

The communication between the RMWCC and the TMWCC could use an interface with similar properties as the S11 interface between MME and the respective Serving Gateway (SGW).

Upon network association of the UE the MME extensions could indicate which RMWCC the UE has to report to, similar to the mechanism used to indicate the responsible SGW.

In a 2G/3G context the SGSN responsible for the UEs of a specific region can re-route the detection traffic to the regional RMWCC.

4.3.2.2 Malware Collector/Controller Hierarchy

As previously explained, we assume there are several Regional Malware Controller / Collector (RMWCC) distributed in the MNO's domain, while there is also one central Top Level MWCC (TMWCC).
RMWCC interact with UEs that are managed by different regional authorities in the current mobility domain of the UE. For instance, a regional MWCC could be associated to a MSC and as such being responsible for the respectively managed UEs. RMWCC collect the events generated by UEs. Based on the event, the RMWCC indirectly queries the TMWCC for further interactions.

In case the TMWCC indicates that a specific detection event is unique for specific UE (class), it commands the RMWCC to obtain additional information from the UE, i.e., a sample of the app suspected to be malware, along with the system call trace that triggered the detection mechanism.

**Figure 21: Anomaly Detection Setup**

The TMWCC is also responsible for triggering reactive measures in the MNO domain once it becomes necessary, e.g., to isolate an infected UE from spreading its infectious software to other vulnerable UEs.

Additionally, the TMWCC can distribute updates of the detection models to the RMWCC for the respective UEs. Managing and obtaining the updated detection models is out of scope of this document. However, below we consider strategies that can aid such functionality.

**4.3.2.3 Network Level Strategies**

In this section we discuss strategies that are prospectively improving the anomaly detection mechanism at network scale.
4.3.2.3.1 Situational Awareness

The MNO is in the position to leverage accessing information about significant amount of users with a heterogeneous set of UEs. This can be used to further improve the efficiency of protection measures not only on a single UE, but also to protect the network.

4.3.2.3.1.1 Model Aggregation

Operating in a MNO setting may allow extending the modeling approach by aggregation and recombination of several models of the same UE. Collecting trace data from devices and training the models for the respective class of UEs can be done in the MNO's domain. Both end user and MNO can profit from this, since the detection accuracy can be increased and the protection and the associated availability of service will increase by the cost of providing the data and entailing the models. Improved models can be used by UEs via the RMWCC. Thus the detection models could be iteratively and incrementally improved covering all design and architecture characteristics like devices, version or operating systems.

4.3.2.3.1.2 Split-Monitoring

The monitoring process on the UEs could also be scaled down and distributed to reduce redundancy. For instance probing could be used to scale down the level of effort, e.g., the same app would be monitored by many UEs, likely even by UEs with the same properties, i.e., hardware, operating system version, etc. However, even though the redundancy is reduced, determining which apps are potentially malicious can still be achieved.

In the unified view this approach would still cover all apps the need to be monitored according to the relevant selection.

4.4 Interfacing FC-IP with FC-MA

The previous sections describe how (from within FC-IP) ‘messages’ will be generated, aggregated and propagated through the network. Sources of messages are in first place the methods running in NE or UE or in some cases also the network-sided management and control systems. A summary referring to the various messages themselves has been presented in Section 3.3. In ASMONIA the transition from message collection to message evaluation is handled with FC-MA, where Figure 22 provides a collocating view, showing how all the end-points and network-sided entities cooperate with each other.

Below we concentrate on a method to integrate all the collected data allowing an overall analysis and assessment. Thus, determining a ‘sanity state’ of the network is derived from an overall view on the plurality of collected information.

4.4.1 Integration of Integrity Observations

We describe an interface definition for the situational picture of the protected critical infrastructure. Due to the high variability of configurations a lightweight data design is provided as a rational scheme enabling to exchange and analyze relevant information provided by the integrity observation system(s). Furthermore analysis is foreseen by the situational awareness information system based on observations to recommend instrumentation of this guard leveraging a wider oversight. This could be successful detections and instrumentations, i.e. relevant components with high contribution in the mapping from the integrity measurement space into the deficit domain, see [ASMONIA_D41ii]. Both in [ASMONIA_D41ii] and [ASMONIA_D42], we proposed data integration techniques for aggregating and combining data, e.g. measurements providing a unified view on these data.

We assume a database, i.e. an organized collection of data with authorization control and sufficient privileges that is maintained by suitable constraints and triggers. We assume a schema as structural description of relations. That means a database instance is an actual
contents at given point in time. As outlined in [ASMONIA_D42] we treat data conceptually relying on the mature theory of relational algebra.

The algebra allows modeling configurations and expressing observations. As such it provides an extensible frame for relevant information. Although there are complex star schemes possible, we assume in [ASMONIA_D42] for illustration purposes and for the reader's convenience the following simple scheme:

- **Invocation** (invocation, agents, observations)
- **Agent** (agent, service, value, cost, deficit)
- **Service** (service, service_description)
- **Observation** (observation, guard, measurement)
- **Guard** (guard, guard_description, measurement_description)

The description of the concepts guard and observation are provided in [ASMONIA_D41ii].

The observation and the guard relation constitute an interface where the different data formats e.g. in XML or JSON can be stored and retrieved as records. The measurements provided by this integrity protecting system are based on configuration information. This can be modeled as ensembles of elements, as outlined in [ASMONIA_D41i] according to the following simplified scheme:

- **Measurement** (part, property, context)
- **Part_of** (part, part)
- **Properties** (property, value)

Measurement is a relation that reveals properties including integrity information about the observed part in a specified context. Integrity information comprises at least “is integer” or “is not integer”. To each part a guard provides this information which is itself a part. The context comprises at least a time. It could also be a location. Context types could be modeled in a separate relation.

Each part could constitute of other parts recursively in a mereological sense.

Whenever a measurement tuple is exists then there is a guard reporting the properties of the observed part. Such properties could be vendor, technology, size, and especially integrity state.

To leverage integrity measurements within the Observation relation, there is an interface to the simplified schema necessary. Therefore we interpret the attribute measurement as the tuple of the measurement relation, namely (part, property, context).

A guard is part of the infrastructure that can be described using the guard relation, i.e. the attribute guard will have the value of this part. The measurement description will be the valid properties and the guard description is an attribute that could be used to identify measurement or classification algorithms. The schema extension can be used to optimize and instrument efficient integrity observations. That would allow to exchange information about which implementation and instrumentation is effective and which configuration is ineffective in contributing to risk reduction.

### 4.5 Overall View

This section shows the complete network-sided security infrastructure for FC-IP in an overall architectural view. In Figure 22 the various entities are assembled, which are necessary to generate, collect and propagate SW-IP and MWD related messages through the network towards FC-MA.
SW-IP and MW related messaging starts from the network nodes, which are either UE or NE. For NE the existing OAM infrastructure can be extended - there may be SW changes, but in standardized parts no new HW or interfaces need to be implemented, but the ASMONIA interfaces towards FC-MA. In contrast, aggregating and propagating messages from UE, as well as trust and SW management of the underlying protection methods, require additional entities and interfaces that in 3GPP compliant ways have to be integrated into a 4G mobile network as described in the sections above.
5 Re-Evaluation Aspects

The following section re-evaluates the developed protection methods, comparing these with initial expectations and showing possible constraints and trade-offs, which in some cases cannot be solved to the favor of ‘highest possible security’.

5.1.1 UE protection concepts

This section concentrates on the re-evaluation of UE-related protection concepts and methods as introduced in ASMONIA. Specifically, the re-evaluation is a critical retrospection and re-discussion referencing to the security requirements mentioned in [ASMONIA_D21] and the resulting methods as explained in [ASMONIA_D22].

5.1.1.1 UE protection methods related to SW-IP

Re-evaluation of integrity protection mechanisms was also part of the previous ASMONIA deliverable [ASMONIA_D22] and in the corresponding papers [Wessel2012, Wessel2013].

We now analyze the security of our proposed remote attestation protocol for mobile baseband stacks, which was first published in [WWS2012]. We mainly consider software attacks, whereas hardware attacks, such as TPM cold boot attacks, are by nature and definition out of scope. As most existing protocols [Yingyou2010], we start from the premise (for the sake of simplicity) that the platform configuration reflects the actual state of the baseband stack at any time. We also assume that it is not possible to forge a trusted platform configuration by exploiting bugs, such as buffer overflows, although that requires either a periodical or an on-demand measurement architecture, such as IBM’s IMA [Sailer2004].

In our first scenario, the attacker attempts to extract the cryptographic keys. However, that is not possible, because the symmetric keys $K^i$, $K^{ATT}$, and the authentication data $A_{\text{wrap}}$ securely stored inside the USIM. The asymmetric keys $K_{\text{wrap}}, K_{\text{seal}}$ non-migratable, thus never leave the MTM.

The attacker could also try to replace the sealed or wrapped keys, namely $\{\text{pkSigBV}\}^{P_c, pk}$ $\{K^{\text{INT}}\}^{P_g, pk}$. In the first case, the sealed public signing key can only be unsealed while the boot loader is executed and the $P_c$ not yet invalidated by $P_B$. Since the boot loader is stored in ROM, it is always executed first and cannot be modified. As a result, the adversary cannot seal a different public key, which would successfully verify a signature for a manipulated baseband update. In the second case, the attacker might try to wrap an integrity key to an insecure platform configuration, e.g., no PCR selected, to manipulate the baseband stack without the attestation protocol noticing. However, this is not possible, because the authentication data $A_{\text{wrap}}$ stored inside the USIM.

In the next scenario, the adversary actually manipulates the baseband binary to attack the network. However, since the baseband binary is measured by the boot loader before it is executed, the manipulation is reflected in the platform configuration $P^{\prime}_c$. As a result, the MTM cannot load the wrapped key, the attestation fails (because of the return code), and the USIM denies access to attestation key $K^{ATT}$. That mean the attestation value $ATT$ cannot be calculated correctly and the network only allows fail-safe access to network, which can effectively prevent the attack. In the case, where the attacker manipulates the baseband binary, but replays an old MTM result message in order to make the USIM believe that loading the integrity key was successful, the USIM simply needs to check the nonceOdd. Since this value is random and only known to the USIM, the attestation fails, because the replayed MTM result message has a different nonceOdd.
In our last scenario, the adversary might try to forge the attestation value $ATT$ in order to access and attack the network with a compromised baseband stack. However, if the attacker is able to capture the random number $RAND$, the authentication token $AUTN$, and the authentication response $RES$, it is still not possible to calculate the correct attestation value. The attacker has no knowledge about the attestation key $K^{ATT}$, which is securely stored in the USIM. Even in the case, where the adversary combines the authentication response $RES$ with an old attestation value $ATT'$, the network only grants limited access. Since the attestation value $ATT$ is an HMAC over the current random number $RAND$, the pre-calculated attestation value $XATT$ does not match $ATT'$, so the attestation fails. The network only allows fail-safe access and an attack on the critical network components, such as the HSS, is prevented.

5.1.2 NE protection concepts

In this section we reconsider the protection paradigms and methods for NE, reconsidering the initial security requirements, as stated in [ASMONIA_D21]. These requirements have been derived from the threat and risk analysis done in ASMONIA [ASMONIA_D51], from security requirements on system integrity coming from 3GPP SA3 (i.e., for Node-Bs/eNB in [TS_33401] and HeNB in [TS_33320]), as well as from additional requirements reflecting security needs of specific implementations and those arising from the long-term and continuous operation and very long life-cycle of NE equipment, which may last over 20 years.

5.1.2.1 Reconsidering the Methods Applied to NEs

Citing [ASMONIA_D51], which also emphasizes insider risks,

```
...to prevent malicious network operator staff from changing the functions of network elements by modifying the software on a network element, e.g. install additional programs, software integrity protection mechanisms can be used...
```

the protection paradigms selected for NEs (remotely signed signature objects) are generally taking insider risks in operator network into account, as modification of signed data and SW is (cryptographically) prevented outside the production environment.

Thus, the fundamental requirements [ASMONIA_D21], RQ-SW-IP -Fundamental-1 on

- Protection of (static) integrity
- Protection of authenticity
- Protecting scope and purpose of SW
- Protecting directives to be enforced for / during / after SW usage (e.g., for invalidation or revocation)
Protection of dynamic integrity (i.e., during operation))

are inherently addressed by using Signed Objects (which are secured independently from the SW-IP use case and media they are stored on, and are self-describing and verifiable from everybody within a given validity period), but in addition may need specific mechanisms for secure implementation. Secure implementation is required in particular, if the verifying and policy-enforcing system itself is threatened and thus must ensure a defined level of attack resistance.

As expressed by RQ-SW-IP-Fundamental-2, special emphasis has to be put on the SW and data, distributed to many (equal or similar) systems as modifications potentially affect all systems, without the need for an attacker to directly access the systems in question themselves. Also in this situation the selected protection paradigm is perfectly suited, as it aims to reduce the attack surface to the production environment, where the protection mechanism is applied, as well as to the verifying target system (which may be in the unsecure domain). Any attack 'in the middle' will reliably be excluded at any time (except in cases, where SW updates, invalidations or revocations are suppressed by an attacker – but even this will be detected after some delay, as soon as a controlling or auditing system inspects an individual NE).

Requirements on support for efficient crypto management and long product life-cycles, as stated in RQ-SW-IP-ASMONIA-3, conceptually can be fulfilled by the preferred PKI based signature scheme, providing proper cryptography (RSA with 2048 bit, hashes built with SHA2 family or better, ..) and appropriate trust management, as enabled with X.509 certificates, associated signatures over data and SW, and related control mechanisms (revocation, exchange of root CA certificates, expiry control, etc.). According principles have been described in [ASMONIA_D22], e.g., in Sections 3.5.2.1.4, 3.7.2.7, 3.7.2.8, and have earlier been motivated in [ASMONIA_D21], Section 3.2.2. and 3.2.4.2.

Considering unintentional modifications, in [ASMONIA_D21], Section 5.3.2.2, generic use cases have been mentioned, where integrity protection has to be applied:

- During development process  
  (meaning that in this phase a reliable and correct cryptographic reference is built)
- During SW delivery
- Before installation (after delivery…)
- While SW is stored (after delivery…)
- While SW is stored (after delivery…)

Except for the first one (during development process), unintentional modifications can be easily parried by a cryptographic protection paradigm. Special care has to be taken during the development process, but partly this falls out of the cryptographic protection paradigm (which is only used to create a cryptographic reference) and instead is subject of a proper and secure SW development process (to give an example: as we are talking about unintentional modifications, it may accidently happen that some faulty test SW was signed, instead of the final release. But this to control is completely up to the manufacturer's accuracy and actually not an issue of the protection mechanism once it is applied. Similarly it is an obliga-

11 Stronger mechanisms for SW integrity protection shall be applied if an attack could
- affects many systems (without individualized manipulations per system)
- be performed without system access (physically or remote), i.e., by modifying SW which is targeted to a system (e.g., on USB stick)
tion for the final SW provider to inspect and test any third party SW before it is used and signed together with its own SW). However, after delivery it falls into the responsibility of a target system to reliably verify, if SW or data has been modified. This seems to be not problematic, as we do not expect targeted attacks in this case (unintentional modifications), but e.g., threats like virus infections, which are not designed to smart out a properly designed SW signing/verification mechanism.

So far, above requirements are not yet demanding very specific and attack resistant implementation mechanisms, taking targeted attacks (against the protection paradigms themselves) into account.

However, resistance against targeted attacks is clearly demanded if we take intentional modifications into account ([ASMONIA_D21], Section 5.3.2.1) and in particular, RQ-3GPP-ASMONIA-1, stating:

Requirements on SW integrity protection in [TS_33401] clause 5.3 (in particular 5.3.2 and 5.3.5) shall be fulfilled for eNBs, e.g., demanding for authorized, integrity protected software during transfer and in particular protection regarding the boot process, but potentially also concerning executed functions of the secure environment (to protect sensitive data at any time))

as well as RQ_3GPP_ASMONIA_2, stating:

For HeNBs RQ_3GPP_ASMONIA_1 has to be respected as underlying set of requirements, which is valid for all types of eNBs. In addition, requirements on SW integrity protection in [TS33320] (in particular clauses 5.1.2, 6.1, 7.1, 8.3.2.2, and 8.4) shall be fulfilled for HeNBs, e.g. on TrE, on integrity protected SW download via TR.069, on secure boot and on autonomous validation.

In addition to the general protection given by the cryptographic paradigm, the fundamental protection developed in [ASMONIA_D22] aims to provide

- Protection for the boot process, including physical access during attacks. Compare [ASMONIA_D22], Section 3.5.2.1.4, where a novel hardware concept has been proposed, which is specifically suited to protect firmware and any modification of firmware by encapsulation of security critical flash memory (e.g., booted SW) into authorization hardware, not controllable via system CPU/SW attack. In addition, higher integration provides protection against (more intelligent) physical attacks.

- Requirements on ‘autonomous validation (only connecting to NW on successful validation)’ thereby is fully supported. In addition, the NW can retrieve an NE SW-update state (in particular: for the HeNB case) at any time via a secured protocol. This closes a gap, which has not specifically been mentioned by the standardization ([TS_33320]), where a HeNB would also connect to the NW with some ‘old’ (i.e., potentially vulnerable, but previously authorized) SW, without notifying its actual SW state in reliable manner. Following the proposed solution the SW state is made visible and an attacker could not keep a system in a vulnerable state. Further the solution enables enforcing the SW update, as it is executed autonomously on explicit remote challenge.

- As already mentioned, long-term scenarios can be managed securely, as any trust management operation (such as e.g., exchange of trust anchor, in case the Root CA has to change its root CA key, or revocation of old SW/FW, preventing from re-flashing vulnerable, but authorized SW) is also supported.

These security features have been demanded by the requirement SW-SW-IP-Verification-Protection (see [ASMONIA_D21]), stating

Security of a SW-IP supporting infrastructure for SW protection shall provide efficient prevention against attacks aiming at
- Manipulating or de-activating verification mechanisms in local or remote system (see Section
5.3.5.2.1)

- (Re -) Installing protected, but vulnerable SW (see Section 5.3.5.2.2)
- Preventing a system from updating to new version (see Section 5.3.5.2.3)

Note that such requirements can only reliably be fulfilled by HW, but if we consider systems with a lower risk (e.g., not prone to physical attacks) such principles could also be implemented via appropriate system FW or kernel module or Hypervisor based SW solution. This depends on the risk level of an individual system and the specific attacks it is exposed to.

Refinement of the AFCP protocol

Reconsidering the HW based memory control process, one issue needs to be refined regarding implementation of the communication protocol as used in [ASMONIA_D22], Section 3.7.2.4: It is mentioned there:

Trustworthiness is important to prevent Man in the Middle (MITM) attacks, e.g., trying to re-send and resign authorized messages, if this is not wanted (by OAM)

However, the protocol introduced in this section may not fully comply with this requirement, particularly if the term 'nonce' is interpreted as an independent, random parameter. In this case only the OAM Server would detect a replay attack, but not the AFCP. Even if this was not the intention, in most cases this is not critical, as it must be seen in combination with the internal logic of the AFCP: For instance, downgrading to old, but signed SW (initiated by replaying an 'old' signed command) would only be possible if this is not prevented by a revocation policy (meanwhile stored in the AFCP), which would not be overwritten. Similarly, 'downgrading' to an old root CA certificate would never be possible, as the mandatory protocol given in [ASMONIA_D22], Section 3.7.2.7, would prevent this. Also, it would not harm to re-send an authorized command requesting to report the integrity state of the flash-memory. But there may be critical commands as well, such as deletion of keys.

In contrast, freshness of commands sent from an OAM server could always be verified by the server, thus a MITM could never fake the AFCP's answer (specifically its internal verification result, when reported on request).

Nevertheless, for a correct implementation, also the AFCP should always be able to proof newness when receiving an authorized command. Thus, we substitute 'nonce' with 'new-nonce (n-nonce)' as shown in Figure 23 below. 'New-nonce' means that the AFCP is enabled to reliably detect re-used nonces. This can be accomplished in several ways:

- The ACFP 'remembers' all nonces used in the past and invalidates them, i.e., rejects all new commands containing such invalid nonce. To implement this, the AFCP could either store and compare the nonce history (which is quite cumbersome) or use an algorithmic approach like the following: We simply construct an n-nonce by concatenating a monotone counter value (such as a time stamp) with a random nonce (i.e., <n-nonce=mctr | random_nonce>). The AFCP only stores the monotone counter value, which must be increased for each 'new' command.

- A second option would be to extend the protocol by an additional round-trip, where the n-nonce is defined by the AFCP, as shown below. Also in this case any replay of old commands could be detected.

- A third option (not shown in the figure) would be to execute commands only on receipt of a closing answer signed and sent back from OAM server, confirming the first response. In this case the internal logic of the AFCP slightly needs to be changed, executing the steps: receipt of command - command verification – sending RES to indicate that the command is understood and the AFCP is prepared for execution – receipt of the closing ok (signed and send by OAM) – execute the
prepared command. With RES the AFCP would need to send (and also sign) an own new nonce\textsubscript{AFCP}, which could be concatenated with the one previously received from OAM. If this resulting n-nonce* is contained in the closing message sent from OAM, the AFCP would be able to proof freshness.

![Diagram](image)

**Figure 23: Refining Figure 31 in [ASMONIA_D22]**

In order to minimize round-trips the first option seems to be most suited. But it is matter of implementation, which solution to select (or even to apply another method for replay protection, not mentioned above).

With this clarification it should now be better defined how to realize a secure and mutually replay protected communication.

So-called ‘Governmental Requirements’, such as stated in [ASMONIA_D21], Section 5.3.4, RQ-SW-IP-Governmental-1 and RQ-SW-IP-Governmental-2 seem to be less stringent and are supposed to be fulfilled, when applying the selected protection paradigms (SW Signing) in an accurate manner. Essentially the ‘SW delivery’ and ‘installation’ use cases are addressed, which could be realized by applying static integrity protection only (i.e., without runtime validation). As it is recommended

’sensitive systems should be protected by use of digital signatures under responsibility of the SW or system manufacturer’ (Legal Interception)

and

‘The Vendor shall maintain integrity of the software build including upgrades, operating systems and application from factory to desk’ (TSP templates, India),

the selected, primarily vendor governed paradigmatic approach is fully in line with such expectations.

In the following we consider attacks during runtime including those targeted against the SW-IP mechanisms implemented within the target systems themselves.

Later, re-evaluation of requirements and methods related to the security infrastructure will be considered separately and summarized in Section 5.1.2.2.

Regarding attacks against target systems, generally we have to consider two different types of potential attacks: Those tried remotely via SW mechanisms (typically, such as an exploit) and those tried by an attacker, who has local physical access to a system. Related require-
ments have been introduced in [ASMONIA_D21], Section 5.3.5.3. The following relevant groups of attacks have been considered:

- Attacker Succeeds to Launch Application Level Exploit (at runtime), Section 5.3.5.3.1
- Attacker Succeeds to Launch Kernel Level Exploit (at runtime), Section 5.3.5.3.2
- Attacker Succeeds to exchange SW in file system, Section 5.3.4.3.3
- Attacker Succeeds to inject code into system memory, Section 5.3.4.3.4
- Attacker Succeeds to Surmount Local Protection by Physical Access, Section 5.3.4.3.5

These risks led to six requirements, RQ-SW-IP-Implementation-1 to RQ-SW-IP-Implementation-6, which actually demand SW-IP mechanisms to implement so that these are protected against exploits and other SW attacks as good as possible, e.g. in kernel or a hypervisor, and are immune against modification of SW in file system, but also in memory.

It is evident that (cryptographic) SW integrity protection alone is never sufficient to parry all the mentioned attack risks, in particular as these also exploit dynamic data and code, whose correctness (i.e., expected values) cannot be pre-calculated or described by cryptographic parameters (e.g., hash values over the stack or heap area). Thus, cryptographic SW-IP must be combined with hardening methods, whose prevailing purpose (in the context of this work) is to protect SW-IP methods themselves from being modified.

Several hardening approaches have been considered in [ASMONIA_D22], Sections 3.7.2.1 – 3.7.2.3, and 3.5.3.1.3, including

- Existing ‘Traditional hardening methods’, such as StackGuard, PAX or GrSecurity, mainly aiming to prevent execution of code in heap and/or stack area.
- Preventive approaches, such as static Code Analysis, aiming to avoid unintentional implementation of vulnerabilities (exploit enabler functionality).
- A specific Security Hypervisor (SecVisor), aiming to prevent code injection into the kernel memory, also including control over DMA.
- MAC approaches to mitigate effects of potentially successful attacks against SW-IP mechanisms and data in kernel area.
- Virtualization as a separate concept to isolate SW-IP mechanisms from direct attacks (assuming essentially the guests are offering attack surfaces).
- Host Based Intrusion Detection Systems (HIDS) offer another option for integrity protection, in particular when combined with system specific log file analysis, but such solutions have not been considered in the context of this work. Mostly, the existing approaches are based on dedicated client server architectures that have to be locally administered. Integration into virtualization concepts as well as adapted self-protection concepts usually are not in focus. Therefore, they do not well match the vendor-governed responsibility and the autonomous validation scope, as applied for NEs. Nevertheless, for NW elements, which are built on standard Linux distribution, HIDS may be applicable with low efforts.

A general difficulty is to evaluate the efficiency of an individual hardening approach or a combination of some of them. This depends from each specific system, the individual SW and operating system chosen, but also from the ‘patch level’, which is controlled by system specific security administration. Nevertheless, balancing efforts and achievable security level, existing kernel controlled MAC approaches (e.g., SeLinux or RSBAC) as well as (well established) virtualization schemes are supposed to offer efficient and economic solutions to reduce the risks down to an acceptable level, without running into a situation where performance degradations and system stability are affected too much. On the other hand counter-
measures, such as provided by SecVisor, seem very promising, but require specific efforts for implementation and exhaustive testing of each specific system to limit possible effects on system reliability and performance. Thus, in practice such an approach may be applied only to a number of smaller, dedicated systems (e.g., an embedded system, such as a hardened HeNB), where very high security requirements are in foreground. For NE-systems, which essentially are built upon standard distributions, and kernels (which may change very often), and possibly already use virtualization approaches for other purposes than security, such a hypervisor solution seems to be too challenging.

None of these hardening mechanisms provides protection against malicious physical access or hostile administration; any bare abused administrator account may be sufficient, to change the system settings so that countermeasures are outfoxed or switched off. Better protection is only possible with separated, remotely controlled HW, such as considered earlier ([ASMONIA_D22], Section 3.5.2.1.4). For an embedded system protected with the method described (holding its hardened FW and SW in validated and write protected flash memory), profound physical modifications would be necessary to change the protected PROM content or to influence its validation. This may not be impossible, but would require significant individual efforts (per system!) and in most cases would leave detectable traces. Again, this reflects the requirements RQ-SW-IP-Implementation-4, RQ-SW-IP-Implementation-5, and RQ-SW-IP-Implementation-6, given in [ASMONIA_D21]. Of course the strength of HW protection depends on the level of integration (FW in PROM/ROM .. resin-cast compound module .. multi-chip module .. ASIC) and the possibilities offered for a specific system design. Thus, in practice the achievable security level will be significantly influenced by (HW-) design decisions, which often are not taken by security considerations alone.

Still, residual risks remain, which cannot be prevented, e.g., hidden backdoors, intentionally implanted vulnerabilities, open interfaces and hidden functionality to download and execute unauthorized code at runtime. In this work, only the last issue is considered, reflecting RQ-RQ-SW-IP-RunTime-1 in [ASMONIA_D21]. The other risks mentioned are subject to a preventive, secure SW development process and therefore, fall out of the scope of SW-IP methods this work is focusing on.

In [ASMONIA_D22] several methods have been described to implement runtime integrity protection: Primarily in scope for NE is event triggered file based integrity protection at load time: Section 3.5.3.1 explains the methods developed. The fundamental approach is implemented in kernel, using LSM hooks (or similar kernel hook, such as coming with RSBAC) initiating file based verification as soon as a binary file is mapped into memory. Depending on the used hook, read access to a file can also become intercepted, offering the possibility of validating any file type, e.g., scripts or configuration files. Access to resources may be blocked (or even only logged) as soon as verification fails. Associated signatures and policies (what to do with a signed file, what to do with unsigned files,..) are looked up using a (externally) signed database, which can also be hold in kernel memory. The same principle can be applied for a Hypervisor (HV), enabling protection of resources (via NFS) given to the virtual machines on top (guests).

When considering the virtualized approach trade-offs becomes visible: While the SW-IP methods and data (verification cache, verification parameters) as such are better protected (within HV kernel), and do not come along with modification in guest, semantic interpretation of what operation is intended (on the file read) by the guest is not longer available. I.e. the HV does not know, whether the guest only reads some binary file or intends to execute it. Moreover, it does not see access to files residing in the guest's file cache. If the guest is vulnerable or compromised by an administrator the attacker would also have a chance to cir-
cumvert the HV based protection, e.g. by either creating binaries in memory (disk), or by manipulating the target IP address of the NFS server. This could be prevented by adaptations to the guest or as in KVM/QEMU case by implementing filter functions into QEMU, respectively the emulated LAN interface. Possible implementation restrictions come with solutions, which intend to use encrypted NFS channel to a remote NFS server. Such approach would only allow verification of the remote file resources, if the HV terminates the security relation but NOT the guest. Thus, specific attention has to be paid when setting up such a HV based integrity protection. If adaptation of guests does not matter, of course it is possible to avoid such limitations, but move the attack surface also back to the guest and thus, into the user space (where a guest is running).

Another interesting approach (page-based verification) for run-time protection has been developed for the protection of UE, but is also based on a Linux system. It is described in [ASMONIA_D22], starting Section 4.1.1.1. In principle this approach could also be applied to NEs, in particular to dedicated, embedded NE, such as an HeNB. For complex NE (to a large extent relying on standard Linux tool chain, libraries and adapted cross-compilation environments for sub-boards and specific components such as network processors) the practical use may be limited by a) the limitation to pure binaries, b) the necessary pre-calculation of a very huge number of page-aligned hashes, requiring adaptations of the standard SW development process, and c) the specific use of a virtualization layer (which could conflict with a standard virtualization solution, similar to the SecVisor approach). Another -organizational-drawback could be, that such approach could not generally be applied (e.g., also for “SW Installation”, which usually is file or image based) but had to be developed and well tested in addition and individually for a specific system.

Summarizing the SW-IP methods for NE it is evident, that the technical solution alone in not the only essential aspect. In practice, crucial trade-offs have to be balanced, including

- Functional split between eligible security layers
  (HW vs. SW, HV vs. VM, kernel-space vs. user-space, etc.)
- Reasonable methods mix
  (static SW-IP / cryptography vs. dynamic: hardening, virtualization, sandboxing,...)
- Availability of a solution/HW vs. appropriateness for specific context
- Cost of implementation vs. achievable security level (as set by ’requirements’)
- Re-usability of an approach vs. efficiency

Concluding, in many situations solution finding is a mixture of technical deliberations against economic effects, system evolution, and feature planning. Part of such considerations is the possible impact on the security infrastructure, which we will re-consider in the following section.

5.1.2.2 Reconsidering Security Infrastructure Aspects

Similar to above mentioned requirements for SW-IP methods and paradigms, in [ASMONIA_D21] requirements have been discussed regarding infrastructure aspects. Risks related to the infrastructure for protection have been listed in Section 5.3.5.1.1 (Attacker Succeeds to Outfox PKI or Signing Entities), 5.3.5.1.2 (Attacker Succeeds to Out fox Approval Workflow), 5.3.5.1.3, (Attacker Succeeds to Apply Weak Protection Mechanisms / Algorithms), 5.3.5.1.4 (Attacker Succeeds to Masquerade ID or Role of “authorized source”), leading to requirement RQ-SW-IP-Infrastructure-Protection. Countermeasures against those risks mostly are parried by setting up involved PKI entities, policies and approval- and control workflows in a secure way. However, this to ensure is less a matter of method, than of security best practices and secure SW development processes, which are completely dependent
from a specific manufacturer or even product. It is assumed that required control is state of the art and does not need specific research, rather than an accurate implementation of security principles adapted to a manufacturers SW development environment.

Much more of interest are requirements regarding the infrastructure for verification. These have been described in [ASMONIA_D21], Section 5.3.5.2.1 (Attacker Succeeds to Manipulate or de-activate the Verification Mechanism), 5.3.5.2.2 (Attacker Succeeds to (re-)install Protected, but Vulnerable Software), 5.3.5.2.3 (Attacker Succeeds to Prevent System from Updating to new Version). Partly, these have already been considered above, discussing hardening and HW based protection mechanisms.

It remains to be mentioned that (at infrastructure level) most relevant attacks, which cannot be parried by the verification mechanisms as such, seem to be “suppression” of SW updates, of revocation mechanisms, or of execution of trust management operations, and “disabling” of verification processes (e.g., by circumventing a SW acceptance procedure or by de-installing or compromising a verification module). While the verification methods in principle allow feedback to the network (sending messages on executed operations or verification state), often either these cannot be implemented sufficiently secure (e.g., if additional security HW in not feasible) or such messages may be intercepted and faked (e.g., in cases where these are not signed). That means that residual risks remain in practice mainly due to insufficient strength of implementation security, even if from a conceptual point of view the verification method may be correct. It may happen that implementation security is less stringent than required, either because risk assumptions have been wrong (system was designed, assuming an attacker does never have physical access or assuming it is free of critical vulnerabilities) or because crucial trade-offs cannot be solved to the favor of security. To give an example: In some cases, a manufacturer may decide on bought-in HW, without the possibility to implement security additions. In other cases, performance and stability constraints may lead to situations where run-time protection cannot be implemented. All such conditions cause residual risks, in addition to those which cannot be covered by the methods themselves (as discussed in the previous section).

Another general risk may come from untrustworthy, careless or malicious service staff, having tools and knowledge to manipulate as system so that a specific verification method is disabled or even faked, in order to be prepared for running an attack. These leaks need to be countered by organizational means.
6 Conclusion

The protection methods come along with specific benefits, but also trade-offs, residual risks, and drawbacks that have to be balanced for an individual network or application scenario. In many cases several stakeholders will be involved, each with individual perspectives, expectations, and duties. The stakeholders include vendors, manufacturers (only ‘a few’ for NE, but ‘many’ for UE), operator, user, and service provider (malware analysts, ‘business owner’ providing or requiring highly secure UE). The data integration for integrity observations allows to assess the risk-posture due to integrity effects as well as the relevance, efficiency, and effectiveness of introduced measures.

This allows to ensure stakeholder profits from higher protection of UE and NE as well as from the benefits coming through support of and feedback from collaborative information sharing; in particular the latter is interesting for threatened UE attached to a network or service, which takes advantage of early information about suddenly emerging risks and possibly may also help timely to reveal root causes for attacks, happened in a partner network.

For NE it is assumed that sharing of information is of lower relevance, but for some type of NE it may also be beneficial (e.g., regarding risks emerging from HeNB of a specific model or version). In NE-case SW integrity protection generally and significantly lowers risks for insider threats and for remote SW attacks aiming at persistent modifications of the system SW. It improves the security and trustworthiness of NE products, which are outside the protected domain of an operator, such as HeNB (where an HeNB owner also may be seen as an ‘insider’). Even if SW integrity protection methods (and especially not malware detection) do not directly prevent against physical attacks, hurdles against local misuse can be erected, e.g., limiting the power of an administrator or (UE-) user account. To some extent also physical protection can be realized or improved by careful and security-aware implementation of SW and FW integrity protection methods – which may also imply HW based solutions, as proposed both for UE as well as for NE.

Apart from strengths and weaknesses, the re-evaluation shows the typical trade-offs and thus, also allows setting of preferences for a particular product implementation or scenario. Moreover, the methods can be applied partially, depending on the product landscape and focus of protection. For instance, a manufacturer or operator could set preferences on the 4G radio access network and on most threatened smart-phones and leave other equipment untouched.

On the other side introduction of malware and attack protection and detection methods is demanding investments and organizational efforts including impacts on the infrastructure, which have to be motivated and balanced against the benefits.

In UE case usually this requires agreements and coordination between several stakeholders (manufacturer, operator, malware analysts) and focusing of a defined protection scenario (network access, applications). It has been shown that UE related methods can be integrated into a mobile network – but to the cost of additional infrastructure equipment and management efforts in operator networks. In addition, for efficient control of malware detection (and analysis) external service providers should be involved.

For NE mutual stakeholder dependencies are smaller, at the most involving manufacturers and operators. Nevertheless, in almost any case NE security must be driven by the manufacturer, which also has higher costs for investment in its own infrastructure and for product implementation. Different from UE integration, the impacts on an existing network infrastructure can be kept minimal and are mostly limited to SW adaptations in the proprietary part of OAM systems and their communication hierarchy.
From security point of view the technical mechanisms are desirable, available, and implementable and also impacts on the security infrastructure appear to be manageable. Nevertheless, it remains a matter of motivation and business considerations to which extent a specific stakeholder would be willing to foster product implementation, application and integration into a mobile network.

To a certain extent better protected products lead to competitive advantages and may serve as ‘business enablers’ - and thus are ‘sellable’. Nevertheless, in many cases direct effects on revenues may not be visible in daily operation (i.e., as long as serious attacks remain absent), but investments immediately are paying off when avoiding critical outages and reducing recovery cost, as well as loss of reputation and trust - which then is expected and becomes measurable through ASMONIA’s FC-MA capabilities and through collaborative information sharing.
Establishing UE and NE Protection Methods
Security Infrastructure Integration and Re-Evaluation
D23-1.0

References


Establishing UE and NE Protection Methods
Security Infrastructure Integration and Re-Evaluation
D23-1.0


[tcg_mtm]  Trusted Computing Group (TCG), Mobile Trusted Module (MTM) Specification


[wessel2013]  Sascha Wessel, Frederic Stumpf, Ilja Herdt and Claudia Eckert, Improving Mobile Device Security with Operating System-level...
Establishing UE and NE Protection Methods
Security Infrastructure Integration and Re-Evaluation
D23-1.0


Establishing UE and NE Protection Methods
Security Infrastructure Integration and Re-Evaluation
D23-1.0

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACN</td>
<td>ASMONIA Collaboration Network</td>
</tr>
<tr>
<td>ACS</td>
<td>Auto Configuration Server</td>
</tr>
<tr>
<td>ACGW</td>
<td>ASMONIA Collaboration Gateway</td>
</tr>
<tr>
<td>AFCP</td>
<td>Authorized Flash (memory) Control Process(or)</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>BSON</td>
<td>tbd</td>
</tr>
<tr>
<td>CA</td>
<td>Certification Authority (in PKI context)</td>
</tr>
<tr>
<td>CONV</td>
<td>Entity executing conversion functions (used in this document)</td>
</tr>
<tr>
<td>CPE</td>
<td>Customer Premises Equipment</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRL</td>
<td>Certificate Revocation List</td>
</tr>
<tr>
<td>DM</td>
<td>Domain Manager</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>EM</td>
<td>Element Manager</td>
</tr>
<tr>
<td>eNB</td>
<td>Evolved ‘Node B’ (Base station in LTE network)</td>
</tr>
<tr>
<td>FC-IP</td>
<td>Functional Cluster Integrity Protection (IP)</td>
</tr>
<tr>
<td>FC-MA</td>
<td>Functional Cluster Monitoring and Analysis</td>
</tr>
<tr>
<td>FS</td>
<td>File System</td>
</tr>
<tr>
<td>FW</td>
<td>Firmware</td>
</tr>
<tr>
<td>HIDS</td>
<td>Host Based Intrusion Detection System</td>
</tr>
<tr>
<td>HeMS</td>
<td>HeNB Management System</td>
</tr>
<tr>
<td>HeNB</td>
<td>Home eNode B</td>
</tr>
<tr>
<td>HSS</td>
<td>Home Subscriber Server</td>
</tr>
<tr>
<td>HV</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>IMEI</td>
<td>International Mobile Equipment Identity</td>
</tr>
<tr>
<td>IP</td>
<td>Instruction Pointer</td>
</tr>
<tr>
<td>IP</td>
<td>Integrity Protection</td>
</tr>
<tr>
<td>JIT</td>
<td>Just-In-Time</td>
</tr>
<tr>
<td>JSON</td>
<td>Tbd.</td>
</tr>
<tr>
<td>KVM</td>
<td>Kernel-based Virtual Machine</td>
</tr>
<tr>
<td>LKM</td>
<td>Loadable Kernel Module</td>
</tr>
<tr>
<td>LSM</td>
<td>Linux Security Modules</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base</td>
</tr>
<tr>
<td>MNO</td>
<td>Mobile Network Operator</td>
</tr>
<tr>
<td>MTM</td>
<td>Mobile Trusted Module</td>
</tr>
<tr>
<td>MWD</td>
<td>Malware Detection</td>
</tr>
<tr>
<td>NE</td>
<td>Network Element</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>NM</td>
<td>Network Manager / Management</td>
</tr>
<tr>
<td>NW</td>
<td>Network</td>
</tr>
<tr>
<td>OAM</td>
<td>Operation and Maintenance</td>
</tr>
<tr>
<td>OID</td>
<td>Object Identifier (X.509 certificate element linking to certificate policies)</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PCR</td>
<td>Platform Configuration Register</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PoC</td>
<td>Proof of Concept</td>
</tr>
<tr>
<td>PROM</td>
<td>Programmable Read-Only Memory</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RMWCC</td>
<td>Regional Malware Controller &amp; Collector</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>RSBAC</td>
<td>Rule Set Based Access Control</td>
</tr>
<tr>
<td>SB</td>
<td>Secure Boot</td>
</tr>
<tr>
<td>SGW</td>
<td>Serving Gateway</td>
</tr>
<tr>
<td>SIA</td>
<td>Security Infrastructure Architecture</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SO</td>
<td>Signed Object</td>
</tr>
<tr>
<td>SoC</td>
<td>System on Chip</td>
</tr>
<tr>
<td>S&amp;R</td>
<td>Service and Repair</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>SW-IP / SWIP</td>
<td>SW-Integrity Protection</td>
</tr>
<tr>
<td>TB</td>
<td>Trusted Boot</td>
</tr>
<tr>
<td>TCB</td>
<td>Trusted Computing Base</td>
</tr>
<tr>
<td>TCG</td>
<td>Trusted Computing Group</td>
</tr>
<tr>
<td>TM</td>
<td>Trust Management</td>
</tr>
<tr>
<td>TMWCC</td>
<td>Top-Level Malware Controller &amp; Collector</td>
</tr>
<tr>
<td>TPM</td>
<td>Trusted Platform Module</td>
</tr>
<tr>
<td>TS</td>
<td>Target System / 3GPP: Technical Specification</td>
</tr>
<tr>
<td>TSS</td>
<td>TCG Software Stack Specification</td>
</tr>
<tr>
<td>TTP</td>
<td>Trusted Third Party</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
</tbody>
</table>
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2012-08-06</td>
<td>Initial version</td>
</tr>
<tr>
<td>0.2 – 0.15</td>
<td>2013-05-21</td>
<td>Intermediate draft versions</td>
</tr>
<tr>
<td>0.16</td>
<td>2013-06-03</td>
<td>Review version(s)</td>
</tr>
<tr>
<td>1.0</td>
<td>2013-06-19</td>
<td>Final version</td>
</tr>
</tbody>
</table>