TU Clausthal

# Entwurf eines Instanz-orientierten Modelles zur integeren Berücksichtigung von Softwareupdates im automotiven Kontext

**Dissertation**

zur Erlangung des Doktorgrades

**der Ingenieurwissenschaften**

vorgelegt von

**M. Sc. Nils Böcher**

geb. in Braunschweig

genehmigt von der Fakultät für
Mathematik/Informatik und Maschinenbau der Technischen Universität Clausthal,

Tag der mündlichen Prüfung
14.03.2024

| | |
|---|---|
| Dekan: | Prof. Dr. Jörg P. Müller |
| Vorsitzender der Promotionskommission: | Prof. Thorsten Grosch |
| Betreuer: | Prof. Dr. Andreas Rausch |
| Gutachter: | Prof. Dr. Andreas Rausch |
| Gutachter: | Prof. Dr. Christian Siemers |
| Gutachter: | PD Dr. rer. nat. habil. Christoph Knieke |

# Zusammenfassung

Eine ressourceneffiziente Kreislaufwirtschaft zeichnet sich durch einen möglichst langen Erhalt der Rohstoffe im Produktlebenszyklus eines Produktes aus. Insbesondere gilt dies für die Elektronik. Dazu gehören ebenfalls effiziente Software-Update-Strategien, verbunden mit Wiederaufbereitungsprozessen und der Reparatur von Elektronik. Durch die Digitalisierung erhöht sich die Nutzung der Möglichkeiten zur Wiederverwendbarkeit eines Produktes. Einheitliche Plattformkonzepte und die Entwicklung neuer Geschäftsmodelle unterstützen diese Chancen, bedingen jedoch zusätzliche Anforderungen zur Langzeitkompatibilität von Komponenten, die von heutigen Sicherheitsarchitekturen noch nicht vollständig erfüllt sind. Dies liegt insbesondere daran, dass aus Security- und Safety-Gründen ein fest hinterlegter Schlüssel respektive irreversibel hinterlegter Datensatz im Produkt übergreifend als Verifizierungsinformationen für alle nacheinander auszuführenden Applikationen in Verwendung gebracht wird.

Diese Abhandlung thematisiert die Erarbeitung der Update-Fähigkeit einer signierten Zielapplikation mit neuer, gültiger Autorität für die Wiederverwendbarkeit des Produktes zur Etablierung einer zeitwertgerechten, ressourceneffizienten Kreislaufwirtschaft und trägt zu einer Verbesserung und Aktualisierung der Updatefähigkeiten für eine Service-orientierte Bereitstellung neuer Softwarekomponenten bei.

Der Lösungsansatz besteht darin, zusätzlich signierte Komponenten einzufügen, die innerhalb einer laufenden Instanz für den Secure-Boot-Prozess verwendet werden können. Diese Komponenten bestehen aus weiterführenden Codeblöcken oder statischen Datensätzen, womit sich die Autoritäten austauschen lassen, ohne dass die laufende Applikation angepasst oder einem Update unterzogen werden muss. Auf diese Weise kann, ausgehend von einem einzigen Vertrauensanker („Root of Trust“), im Produkt eine Vertrauenskette („Chain of Trust“) aufgebaut werden, die über eine beliebige Anzahl erneut in Verwendung gebracht werden kann. Zur Absicherung der Sicherheitsanforderungen wurde anhand des Lösungsansatzes eine Risikobewertung erhoben, die an eine Angriffs- und Risikoanalyse nach ISO 21434, ISO 26262, ISO 18045 und SAE J3061 angelehnt ist.

# Abstract

An efficient circular economy is achieved through long-term retainment of the raw materials in the product life cycle. In particular this applies to electronics and encompasses efficient software update strategies essential for effective recycling and repair of electronic devices. The onset of digitizing enabled novel solutions to reuse and recycle existing devices. However, although these solutions are supported by given platform concepts and the development of new business models. The additional requirements for long-term compatibility of components are not yet fully met by current security architectures. Indeed, for security and safety reasons, a permanently stored key or irreversibly stored data record is used across the system as verification means for all applications, which effectively renders device unsuitable for recycling (repurposing) or repair. Therefore, the overall aim of this thesis was to address the issue of reusability of the electronic devices in a current value-based, resource-efficient circular economy without compromising security requirements.

In this thesis the development of the update capability of a signed target application with a new, valid signature source is described. Furthermore, the solution offered here helps to improve the update capabilities with a new, valid signature source for a service-oriented provision of new software components.

This novel approach includes the insert of additional signed components that can be used within a running instance for the secure boot process. These components consist of additional code blocks or static data sets, which allows for an exchange with the signing sources without affecting the running application. In this way, starting from a single trust anchor (“Root

of Trust"), a trust chain ("Chain of Trust") can be built up in the product, which can be used repeatedly. To safeguard the security requirements, a risk assessment of the proposed solution has been conducted based on an attack and risk analysis according to ISO 21434, ISO 26262, ISO 18045 and SAE J3061 .

# Danksagung

Zunächst möchte ich mich bei Prof. Dr. Andreas Rausch bedanken, der mich als externer Doktorand in seinem Institut aufgenommen hat. Seine konstruktive Kritik und leitenden Fragen haben mir geholfen, beginnend mit einem kleinen technischen Problem, den Inhalt dieser Arbeit richtig zu gestalten. Ich möchte mich außerdem bei Dr. Andreas Wenda bedanken, der mich gerade in der Anfangsphase der Promotion begleitet hat und bei Prof. Dr. Christian Siemers und PD Dr. rer. nat. habil. Christoph Knieke für die hilfsbereite und wissenschaftliche Betreuung.

Auch meine Doktorandenkollegen trugen einen großen Teil dazu bei, diese Forschungsarbeit fertigzustellen. Hier möchte ich Dr. Gregor Blott danken, der maßgeblich zu der Umsetzung der Implementierungen beigetragen hat. Markus Bormann danke ich für konstruktive Diskussionen rund um das Thema Nachserienversorgungsstrategien. Ein besonderer Dank gilt auch Jens Reichert, der mir in all den Jahren nicht nur bei Security-Fragen zur Seite stand. Über alle inhaltlichen Aspekte hinaus, bedanke ich mich bei Kriminaloberrat Wolfgang Strehmel und Kriminaldirektor Oliver Mayer, die mir trotz sämtlicher Lagen immer den Rücken freigehalten haben. Insbesondere bedanke ich mich bei meinem familiären und sozialen Umfeld, das dazu beigetragen hat, mich stetig weiterentwickeln zu können. Leipi, Peter, Florian und Christian, danke, dass ihr immer da seid. Meiner Familie danke ich für die Hilfe in allen Lebenslagen. Gerade meine Eltern Christel und Gerhard unterstützen mich vorbehaltlos bis zur Erreichung all meiner Ziele.

Ganz besonders möchte ich mich an dieser Stelle bei meiner Frau Dr. Tijna Alekseeva bedanken, die mich über Jahre hinweg ununterbrochen ermutigt und unterstützt hat. Ihr widme ich diese Arbeit.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung und Motivation

Die Folgen der Komplexitätssteigerungen in der Software sind für die Zuverlässigkeit und Beherrschbarkeit von Prozessen über den Produktlebenszyklus bisher noch nicht vollständig abzusehen. In der Wiederaufbereitung von Produkten, insbesondere im automobilen Bereich, besteht vor diesem Hintergrund noch erhöhter Handlungsbedarf. Hier steigen die Herausforderungen beim Management des Produktanlaufs und der Veränderung dieser Systeme während der Laufzeit über den Produktlebenszyklus. [27]

Ein für die Industrie wichtiger Faktor ist die Wirtschaftlichkeit. Hier müssen stets zeitwertgerechte und kostengünstige Lösungsansätze verfolgt werden. Dies steht jedoch im Konflikt mit dem Endverbraucher, da für den Absatz von Erzeugnissen weitere Aspekte in Betracht gezogen werden müssen. So steigt beispielsweise das Durchschnittsalter der in Deutschland zugelassenen PKWs stetig. Waren es im Jahr 2000 im Durchschnitt noch sieben Jahre, hat sich dieser Wert bis Anfang 2015 auf neun Jahre erhöht. [28] 2019 lag der Durchschnitt bei 9,6 Jahren. [29] Zudem ist die Lebensdauer bis zur Verschrottung (Stand 2014) auf 18 Jahre angestiegen. Für diese Entwicklung liegt das größte Ursachenfeld in der Haltbarkeit. Verbesserte Technologien im Karosseriebau sowie neue Motoren- und Getriebetechnologien erhöhen die Lebensdauer und Laufleistung der Automobile zunehmend. [28]

Markttrends zur Digitalisierung zeigen vermehrt, dass neben dem Produkt auch Servicedienstleistungen im Angebot stehen. Durch die Vernetzung von Geräten miteinander werden nicht nur Systemoptimierungen erzeugt. [30] Die Motivation der Unternehmen zur Investierung in langlebige Produkte mit der Etablierung von ressourceneffizienten Kreisläufen infolge innovativer Geschäftsmodelle, Dienstleistungen und Kundenbindungen ist hier der wichtigste Treiber.

Moderne Konzepte einer ressourceneffizienten Kreislaufwirtschaft verfolgen das Ziel, den Wert von Produkten, Komponenten und Rohstoffen innerhalb der Wirtschaft so lange wie möglich zu erhalten und möglichst wenig Abfall zu erzeugen. [31] Es wird daher erwartet, dass die Elektronik eines Produktes/Systems über den gesamten Produktlebenszyklus funktioniert und die Entscheidung zum Beenden des Lebenszyklus und damit zum Recycling des Produktes/Systems aus anderen Gründen getroffen wird. Dazu können effiziente Softwareupdates im Produktlebenszyklus, verbunden mit Remanufacturing und Reparatur von Elektronik, im Aftermarkt einen Beitrag leisten. [32]

Die Herausforderung für Innovative Produktkreisläufe ist auch in der Förderung von Forschungs- und Entwicklungsvorhaben zum Thema geworden. Die Entwicklung neuer Geschäftsmodelle oder Kooperationsformen zur Umsetzung einer ressourceneffizienten Kreislaufwirtschaft, insbesondere unter Nutzung der Möglichkeiten der Digitalisierung, erhöht die Wiederverwendbarkeit eines Produktes. Dies erfordert jedoch zusätzliche Funktionen und Schnittstellen für digitale Systeme zur Langzeitkompatibilität von Komponenten. Datenmanagement und zuverlässige Datenlöschung von Gebrauchtgeräten könnten ebenfalls bei der Integration nachfolgender

Wertschöpfungsstufen für ein Re-Use berücksichtigt werden. Somit müssen Strategien entwickelt werden, die die Verwendung des Produktes in anderen und zusätzlichen Einsatzgebieten ermöglichen. [33]

Neben der Softwareentwicklung muss sich zwangsläufig auch mit Updates, Patches und dem Schutz der Software befasst werden. Gerade Berechtigungsmanagementkonzepte für den Produkteinsatz und zum Schutze des geistigen Eigentums des Herstellers werden vermehrt eine Rolle in Einsatz- und Ausrollkonzepten spielen. Software-Updates zu aktuellen und künftigen Fahrzeugkonfigurationen bilden dann nicht mehr die Lieferkette eines klassischen, physischen Produktes. Diese Veränderung gewährt vielmehr eine Serie von Nutzungs- und Zugangsrechten, die über die Zeit aktiviert, erneuert und aktualisiert werden. Diese Rechte steuern, welche Funktionalität des Erzeugnisses beim Kauf und bei der Aktivierung bereitgestellt wird.

## 1.1 Ausgangssituation

In einer klassischen Landschaft des Produktentstehungsprozesses bringt ein Lieferant dem Kunden (OEM) ein komplettes System aus Hardware und Software, wie beispielsweise ein Steuergerät. Dieses Produkt wird beim OEM komplett in das Fahrzeug in der jeweiligen topologischen Anordnung in das Endprodukt eingefügt.

In Anbetracht ständig komplexer werdender Domäne-Steuergeräte wird die Hardware von einem Zulieferer (TIER) und die Software, die die Funktionen implementiert, von einem weiteren Zulieferer, einer sogenannten Drittpartei, bereitgestellt.

Moderne Fahrzeuge sind komplexe Systeme, in denen Kommunikationsprotokolle für eine physisch isolierte Netzwerktopologie ausgelegt sind. Im Rahmen steigender Connectivity Services werden vermehrt internetfähige Geräte mit dieser Struktur verbunden. Diese Erhöhung der Konnektivität führt forciert zu neuen Angriffsvektoren und steigert den Anspruch an sicherheitskritische Funktionen des Fahrzeugs.

Diese Technologien können über standardisierte Spezifizierungen neue Geschäftsmodelle im Sinne von Servicedienstleistungen etablieren und die Sicherheit im Safety-Aspekt der Fahrzeuge erhöhen. Hierzu zählen z. B. die europäische eCall-Initiative, der elektronische Fahrtenschreiber EDR oder weitere Service-orientierte Modelle.

Standardmäßig sind etablierte IT-Sicherheitslösungen nicht anwendbar für Fahrzeuge aufgrund von Ressourcenbeschränkungen und Kompatibilitätsproblemen – insbesondere in Anbetracht bestehender Ansätze, Fahrzeuge als monolithische Systeme zu bewerten. Die Leistungen und Kosten der Lösungen können ohne Berücksichtigung des komplexen Lebenszyklus von Automobilkomponenten und des facettenreichen Automobil-Ökosystems nicht bewertet werden, da sie eine große Anzahl von Akteuren umfassen.

Bei Berücksichtigung von Wechselwirkungen unter den verschiedenen Akteuren intensivieren sich weitere Anforderungen an die Architektur von Fahrzeugen. Neben sicheren Kommunikationsprotokollen stehen die Betrachtung der Berechtigungen und die Sicherstellung der Funktionalität über den Produktlebenszyklus im Fokus. Weitere ausführliche Beschreibungen finden sich unter [34], [22], [4], [35] und [36], auf denen folgende Herausforderungen basieren.

## 1.2 Gegenwärtige Herausforderungen

Kryptografische Prozesse zur Lösung sicherheitstechnischer Herausforderungen beeinflussen die unterschiedlichen Auswirkungen auf den Fahrzeuglebenszyklus. So muss z. B. ein Fahrzeughersteller berücksichtigen, dass für die Verwendung sicherer kryptografischer Protokolle zur Kommunikation zum Schutz der Datenintegrität eine korrekte Verwaltung der kryptografischen Schlüssel erforderlich ist, beispielsweise für fertigungstechnische Zwecke oder zum Erhalt des Produktlebenszyklus im Produkt selbst. Dies impliziert auch die Weitergabe wichtiger und vertraulicher Informationen an Lieferanten, Geschäftspartner und andere legitimierte Drittparteien.

Um das Produktmanagement über den Lebenszyklus abbilden zu können, gibt es verschiedene Lösungsaspekte, die jedoch stark Kosten-orientiert, im schlimmsten Fall sogar zu kostspielig sind und nicht eingesetzt werden können.

Sollte ein Steuergerät kompromittiert sein, kann es innerhalb eines Fahrzeuges in die Lage versetzt werden, sicherheitskritische Angriffe auszuführen. Neben der Untergrabung von Fahrzeugfunktionen können im Connectivity-Bereich beispielsweise gekoppelte Geräte oder Backendsysteme Ziel eines Angriffes sein. [37] Wenn eine Lösungsmethodik universell eingesetzt wird, ist daher nicht nur das einzelne Fahrzeug betroffen, sondern im Zweifel auch eine ganze Flotte. Dies hat zudem Auswirkungen auf Änderungen der Garantierückgaben oder Abweichungen von den vorhergesagten Serviceleistungen, beispielsweise bei einem Umstieg von Kommunikationsstandards auf 5G und der damit verbundenen Übergangsphase der Abschaltung des 3G-Netzes. Auch bereits im Einsatz befindliche Fahrzeugflotten benötigen im Zweifel eine neue Telematic Unit, langfristig

durch sich erneuernde Standardisierungen, in jedem Fall aber Softwareupdates.

## 1.3 Zielsetzung

Die Kernmotivation dieser Arbeit liegt in der Erarbeitung eines Konzeptes zur langfristigen Nachserienversorgung im Sinne der Updatefähigkeit einer Zielapplikation. Hierzu steht der Re-Use-Aspekt im Fokus, um moderne Nachserienversorgungsstrategien signifikant zu verbessern. Auch zur Laufzeit des Produktes selbst ist der Aspekt zur Verbesserung und Erneuerung von Updatefähigkeiten für eine Service-orientierte Bereitstellung neuer Softwarekomponenten eine tragende Einflussgröße hinsichtlich des Forschungsbeitrages.

## 1.4 Aufbau der Arbeit

Die Arbeit gliedert sich in sieben Bereiche, von denen neben der Einleitung zwei Grundlagenkapitel vorangestellt werden. Anschließend wird die Forschungsfrage in einem eigenständigen Kapitel erläutert und es folgt im fünften Kapitel das Konzept für die Realisierung der Updatefähigkeit von irreversibel geschützten Applikationen. Anschließend findet eine zugehörige Angriffs- und Risikoanalyse statt. Im siebten Kapitel wird eine Zusammenfassung vorgenommen und ein Ausblick gewährt.

Kapitel zwei stellt die Grundlagen eines Secure-Boot-Prozesses vor. In vier Abschnitten werden Implementierungen und Methoden vorgestellt, die das

Handling eines Secure-Boot-Prozesses beschreiben. Mit inbegriffen sind Erläuterungen für vertrauenswürdige Plattformmodule, die zur Glaubwürdigkeit und Gewährleistung eines Prozesses sowie zur Verbesserung der Sicherheit von Systemplattformen beitragen. Beschrieben sind zudem Validierungsprozesse, welche eine wesentliche Grundlage der weiteren Arbeit bilden.

Das dritte Kapitel beschreibt auf strategischer Ebene den Durchlauf eines Produktes durch den gesamten Produktlebenszyklus (PLC). Separiert und in die jeweiligen Phasen unterteilt werden Grundlagen für die Produktanforderungen, die Serienphase und eine fokussierte Betrachtung der Nachserienversorgungsstrategien sowie deren Lifecycle Managements vorgestellt. Diese Grundlagen werden in die Forschungsfrage aufgenommen und anschließend im Rahmen der Problemstellung beschrieben.

Neben bekannten Sicherheitsanwendungen aus anderen Plattformarchitekturen wird ein detaillierter Durchlauf eines beispielhaften Produktes aus dem Car-Multimedia-Bereich durch den gesamten Lebenszyklus begleitet. Grundlage hierzu bilden die vorangestellten Grundlagenkapitel. So wird im vierten Kapitel die Kernproblematik am Beispiel einer Komponente aus dem Automobilbereich respektive dem Infotainment-Bereich beschrieben. Hier wird detailliert auf die Hardwarearchitektur eingegangen, um mittels der dargestellten Konzepte von Fertigungs- und Wiederaufbereitungsprozessen auf die Erarbeitung der Zielsetzung und somit der Forschungsfrage hinzuwirken, die im letzten Abschnitt dieses Kapitels vorgestellt wird.

Das fünfte Kapitel erarbeitet ein Konzept für die Realisierung der Updatefähigkeit von irreversibel geschützten Applikationen mit neuer Autorität. Im Kern steht die Betrachtung der Fragestellung aus Kapitel 4 die Benennung der Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer, gültiger Autorität zur Verbesserung des Produktlebenszyklus

sowie der Lösungsansatz, zusätzliche Komponenten einfügen zu können. In Anlehnung an das erste Grundlagenkapitel werden Signierungs- sowie Validierungsprozesse aufgegriffen und der Software-Integrationstests vorgestellt. Weiterführend wird ein sich aus der Fragestellung ergebenes Basismodell erstellt und folgend die Lösungsmethode aufgesetzt, die neben der Updatefähigkeit auch eine serviceorientierte Bereitstellung neuer Softwarekomponenten nachfolgender Instanzen sicherstellt. Durch die Betrachtung künftiger E/E-Architekturen kommt es auch zu neuen Partitionierungen im System. Hierzu wird im letzten Abschnitt ein Bezug zum erweiterten Basismodell erstellt.

Um sicherzustellen, dass die vorangestellten Lösungsaspekte im Kapitel 5 die in Kapitel 4 gesetzten Ziele erreichen, müssen diese nachweislich evaluiert werden. Hierzu wird im sechsten Kapitel anhand des Leitbeispiels eine Angriffs- und Risikoanalyse erstellt, die nach ISO 26262, ISO 18045 und SAE J3061 für die Methodik zur Evaluation von IT-Sicherheit ausgerichtet ist. Die Kernbetrachtung dieses Sicherheitskonzeptes liegt in der Evaluierung des in Kapitel 4 erstellten Block-orientierten Modells.

Das siebte Kapitel beinhaltet die Schlussfolgerung der in Kapitel 4 gestellten Forschungsfragen. Weiterführend wird der Forschungsbeitrag mit dem Abgleich der im dritten Abschnitt resümierten Forschungsfragen aus Kapitel 4 zusammengeführt. Final wird eine Abgrenzung beschrieben und ein Ausblick geboten.

# 2 Grundlagen

Zur Aufnahme der eigentlichen Problemstellung und der Forschungsfrage stellt dieses Kapitel Grundlagen der Absicherungsmaßnahmen im Secure-Boot-Verfahren vor.

Der erste Abschnitt beginnt mit grundlegender Begriffsdefinitionen über die Integrität, Authentizität und Verfügbarkeit.

Absicherungsmaßnahmen des Secure-Boot-Prozesses in einem Steuergerät und dessen Systemgrenze werden im zweiten Abschnitt erläutert. Hier werden wesentliche Eigenschaften des Systems vorgestellt und konkret auf Fuse-Register und dem Hardware-Security-Modul eingegangen. Im dritten Abschnitt ist das Secure-Boot-Verfahren und die Secure Execution vorgestellt. Validierungsmethoden der Softwareintegrität, insbesondere dass hier in der Dissertation verwendete signaturbasierte Verfahren, werden im vierten Abschnitt behandelt.

Final wird die in der Arbeit verwendete Taxonomie, insbesondere die Darstellung von abstrakten Codeblöcken, betrachtet. Dabei wird das Memory Mapping beschrieben.

## 2.1 Begriffsdefinitionen

Dieser Abschnitt befasst sich zu Beginn mit den Begrifflichkeiten der Kryptografie. Nach Swoboda et al. [1] sind die Anforderungen als Sicherheitsdienste definiert und in drei Kernbereiche unterteilt, welche als CIA Traid bezeichnet wird und die Anforderungen der Confidentiality (Vertraulichkeit), Integrity (Integrität) und Availability (Verfügbarkeit) beinhaltet.

### 2.1.1 Vertraulichkeit

Die Vertraulichkeit ist ein Dienst mit der Funktion, dass Nachrichten/Daten vor unberechtigtem Zugriff Dritter geschützt sind. Dies betrifft die Sicherheit von gespeicherten Daten und den Informationen bei der Datenübertragung.

### 2.1.2 Integrität

Als Grundlage muss sichergestellt sein, dass eine Nachricht ein Datum unverändert und nicht manipuliert worden ist. Hier werden verschiedene Methoden eingesetzt. Beispielsweise kann ein Geheimnis Verwendung finden, das nur die beiden Kommunikationsteilnehmer kennen.

### 2.1.3 Verfügbarkeit

Die Verfügbarkeit beschreibt die Anforderung, dass Zugriff auf ein System oder Datum gegeben ist.

### 2.1.4 Weiterführende Sicherheitsdienste

Weiterführend werden ableitend nach [1] abgeleitete Sicherheitsdienste definiert und in Abbildung 2.1 veranschaulicht.

#### 2.1.4.1 Authentizität

Authentizität ist eine Sicherheit über die Identität eines Kommunikationspartners. Es muss gewährleistet sein, dass die Herkunft der Information respektive der Nachricht zweifelsfrei zugeordnet werden kann. Eine solche Methodik ist die digitale Signatur, die unter 2.4.3 beschrieben ist.

#### 2.1.4.2 Autorisierung

Neben der Authentizität muss sich zwangsläufig auch mit den Zugangsrechten befasst werden. Dies impliziert die Durchsetzung von Zugriffsrechten als organisatorische Maßnahme, die von einem Managementsystem ausgeht.

#### 2.1.4.3 Verbindlichkeit

Abschließend gilt als Nachweis gegenüber Dritten, wer der Autor einer Nachricht war. Dies beschreibt die Verbindlichkeit.

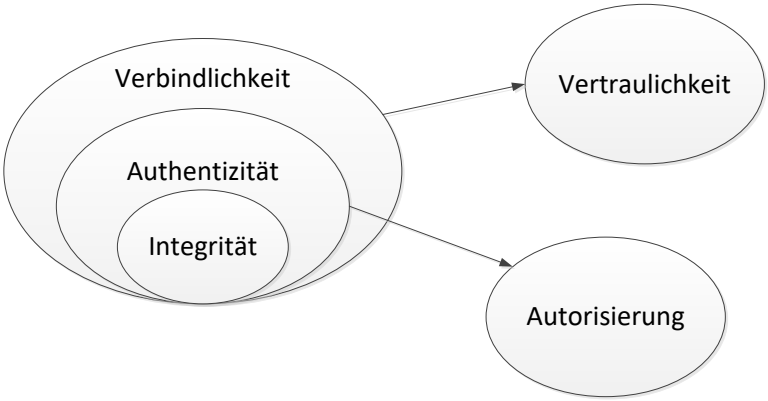


**Abbildung 2.1:** Bezug der Sicherheitsdienste nach [1]

## 2.2 Absicherungsmaßnahmen des Secure-Boot-Prozesses in einem Steuergerät

Nach den Grundlagen der Begriffsdefinitionen im Abschnitt 2.1 werden folgend die Absicherungsmaßnahmen des Secure-Boot-Prozesses in einem Steuergerät vorgestellt.

Zunächst wird auf einen allgemeingültigen Aufbau der Netzwerktopologie von Steuergeräten eingegangen. Aufgrund der damit verbundenen Anforderungen an ein einzelnes Steuergerät soll außerdem eine System-on-Chip-Komponente (SoC) vorgestellt werden, mit der kryptografische Operationen vorgenommen werden können. Die System-on-Chip-Komponente stellt zudem die in der Dissertation betrachtete Systemgrenze dar.

#### 2.2.0.1 Anforderungen an Steuergeräte im Kontext künftiger E/E-Architekturen

Der Aspekt künftiger E/E-Architekturen im automobilen Bereich verfolgt eine zentralisierte Aufbereitung von Informationen mit Fokus auf der Signalverarbeitung. Die heutigen Trends, wie z. B. autonomes Fahren und die damit verbundenen Connectivity Services, deuten auf eine zunehmende Funktionsüberschneidung im Fahrzeug hin. Daher steigt die domänenübergreifende Kommunikation und die Gateway-Last wird höher sein. Derzeitige dezidierte Steuergeräte werden somit künftig zusammengefasst. Dies impliziert auch die Auflösung klassischer Domänen-Bereiche (Powertain, Infotainment etc.) hin zu einer Zonen-orientierten Netzwerktopologie.

Abbildung 2.2 zeigt diesen Transformationsprozess. Weiterhin besteht die Herausforderung darin, im Rahmen der Produktentwicklung den kontinuierlichen Sicherheitsaspekt zu berücksichtigen. [19, 18]

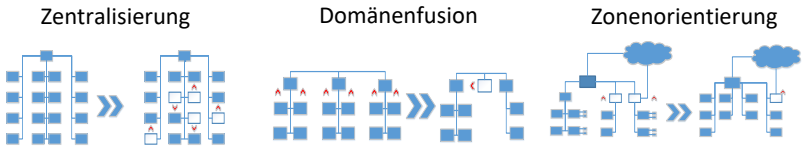


**Abbildung 2.2:** Transformationsprozess: Future-E/E-Architektur nach [2]

#### 2.2.0.2 Allgemeingültiger Aufbau eines Steuergerätes

Die Peripherie von Steuergeräten wird häufig im Konzept mit SoC oder Multiprocessor System-on-Chip (MPSoC) verwendet. Dies gilt insbesondere für Steuergeräte im automobilen Bereich. Durch die Fertigung im Rahmen einer hohen Stückzahl werden Prozessor, Speicher und eine bestimmte Infrastruktur zu einer heterogenen Schaltung zusammengefügt. [38] Auch sind hier Anforderungen bezüglich Safety und Security zu beachten, die konkreter im nachfolgenden Kapitel unter 3.2.1.2 und 3.2.1.3 dargestellt werden.

Im Kern ist ein Steuergerät in drei Bereiche gegliedert, die sich in einer analogen zu digitalen Signalverarbeitung und einer Ansteuerung von Aktorik strukturieren. Beginnend mit der Signalaufbereitung von analogen und digitalen Sensorsignalen steht im Kern der verarbeitende Controller. Dieser steuert und regelt nach der Signalverarbeitung anschließend interne Kommunikationsbusse und Treiberstufen für die Aktorik.

#### 2.2.0.3 Allgemeingültiger Aufbau eines System-on-Chip (SoC)

Die Signalverarbeitung, aber auch die Kommunikation zu anderen Teilnehmern in einem Netzwerk stellt eine Vielzahl von Anforderungen an das SoC.

Neben Funkschnittstellen und Anschlussstellen für externe Kommunikationsbusse werden auch Ports für periphere Anschlüsse bereitgestellt. Im Kern ist hier ebenfalls die klassische Aufteilung der eingangsseitigen Signalverarbeitung durch Kommunikationsschnittstellen, die Datenverarbeitung mittels externer Anschlüsse für Speichermedien und die ausgangsseitigen Ansteuerungsmöglichkeiten für Endgeräte, beispielsweise eines Displays, gegeben. [39]

Speziell der Security-Bereich stellt eine besondere Gewichtung in einem SoC dar.

Im Kern geht es um den zentralen Sicherheitsaspekt bezüglich der Authentifizierung zur Nutzung von Schnittstellen und der gesicherten Bewältigung von kryptografischen Aufgaben. Für die Grundkonfiguration werden Fuse-Register verwendet, die nachfolgend erläutert sind. [40]

### 2.2.1 Fuse-Register: Ein Vertrauensanker

Ein erheblicher Teil der integrierten Systeme innerhalb eines SoC ist dafür vorgesehen, Diagnoseoperationen, Validierungen und Selbsttests zu erledigen. Dies macht sich gerade in der Startsequenz bei der Bestromung eines SoC bemerkbar, wenn Rekonfigurationen, Fehlermanagement und andere Überwachungen Verwendung finden. Um bei der Produktfertigung oder vorab im Entwicklungsprozess bei flexiblen Architekturen einen Zugriff

und besondere Funktionen zu ermöglichen, wurden einheitliche Schnittstellen wie das JTAG (Joint Test Action Group)-Interface nach IEEE 1149.1 standardisiert. Zu der Laufzeit und im regulären Betrieb des SoC im Rahmen des Produktes widerspricht jedoch klar der Sicherheitsaspekt einer Erreichbarkeit solcher Funktionalitäten über eine einheitliche Schnittstelle. Angreifer könnten diese leicht ausnutzen, um Zugriff auf das SoC zu erhalten und beispielsweise an geschützt abgelegte Daten zu gelangen oder sonstigen Einfluss wie Debug-Funktionalitäten auf gesperrte Bereiche auszuüben. Dafür ist es zwingend notwendig, die Zugänglichkeit der Infrastruktur auf das SoC einzuschränken oder ganz zu unterbinden. Gängige Methoden nach [21], [41] und [1], die eine Verschlüsselung oder Verschleierung vorschlagen, erhöhen zwar das Schutzniveau, können jedoch bei einem physikalischen Zugriff einen Angriff nicht vermeiden.

Mit Hilfe der Verwendung von Fuse-Registern können durch das Setzen irreversibler Zustände Konfigurationen des SoC vorgenommen werden. Mittels Auslösen einer On-Chip-Sicherung wird in der elektronischen Schaltung ein Zustand gesetzt, der mit einer verknüpften Logikschaltung eine vorkonfigurierte Eigenschaft auslösen kann. Abbildung 2.3 zeigt die allgemeingültige Darstellung einer Auswerteeinheit nach [3].

Beispielsweise können somit im Rahmen einer Serienfertigung dargestellte Adressbereiche interner Speicher eines SoC einmalig beschrieben werden. Auch kryptografische Schlüssel können daher irreversibel im SoC abgelegt werden [39]. Auch wird es möglich, zusätzliche Konfigurationen wie das Schließen physikalischer Schnittstellen (z. B. JTAG-Schnittstelle) zu kreieren oder Teile davon mit Zugriffsberechtigungen zu versehen. Diese Berechtigungen werden durch eine interne Auswerteeinheit überprüft. Ziel dieses Ansatzes ist es, sicherzustellen, dass nur berechtigte Benutzer einen

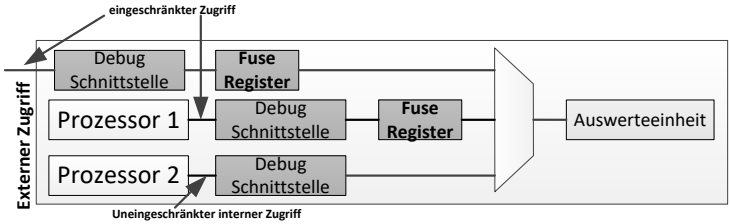


**Abbildung 2.3:** Allgemeingültige Darstellung einer Auswerteinheit nach [3]

Zugang erhalten. Die Authentifizierung kann z.B. über eine Challenge Responce oder über Authentifizierungsschlüssel erfolgen.

### 2.2.2 Hardware-Security-Module

Für den Betrieb sicherer Dienste auf einem Steuergerät und die damit verbundenen begrenzten Ressourcen für Speicherplatz, Prozessorleistungen, aber auch Energiebedarf sind hardwarebasierte Sicherheitslösungen erforderlich. Daher werden Hardware-Security-Module (HSM) verwendet, die den Anforderungen nach Integrität, Authentizität, aber auch zeitlicher Performance genügen. [42]

Die Ausführung kann als vorgefertigter Funktionsblock eines Chipdesigns (IP-Core) im Rahmen eines Coprozessors eines SoC ausgelegt sein oder dediziert als eigenständige integrierte Schaltung (ASIC). Ein HSM zeichnet sich durch einen getrennten Speicherbereich aus, wo kryptografische

Grundelemente bereitgestellt werden. Auch fungiert ein HSM als Hardwarebeschleuniger, der kryptografische Programmierschnittstellen (APIs) beschleunigt. So kann z. B. ein Controller direkt über SPI oder DMA Daten an das HSM-Modul übertragen. Im Mittelpunkt steht der Manipulationsschutz von Rechenoperationen, aber auch der verwendeten Daten. Hier werden die im Rahmen des PKI-Managements eingesetzten kryptografischen Schlüssel gesichert abgelegt. Auch stellt ein HSM gewisse Sicherheitsmaßnahmen bereit, die nur einen autorisierten Betrieb ermöglichen oder das Auslesen seiner Daten erlauben. [43]

Die im HSM abgelegten Schlüssel können für unterschiedliche kryptografische Prozesse verwendet werden. Für das Handling der Schlüssel sind im HSM prinzipiell weitere, von der Applikation unabhängige Schlüssel abgelegt, die den Schutz, Transport und die Authentifizierung der PKI-Schlüssel sicherstellen.

So werden der Transport und allgemein auch die Kommunikation über eine Schnittstelle während eines Schlüsselaustausches verschlüsselt. Funktionsschlüssel werden in bestimmten kryptografischen Funktionen ausgeführt, beispielsweise beim Errechnen im Rahmen von Authentifizierungsmaßnahmen zum HSM.

Der Kernaspekt ist das Speichern der PKI-Schlüssel, welches separat im Anhang befindlichen Abschnitt A.3 erläutert wird. Zum Speichern der PKI-Schlüsel ist in der Regel ein Master Key (MK) in Gebrauch. Der MK ist ein interner, nicht auslesbarer symmetrischer Schlüssel, der die abgelegten PKI-Schlüssel beispielsweise über das AES-Protokoll verschlüsselt ablegt. Die Ablage kann innerhalb von HSM oder außerhalb auf einem externen Speicher erfolgen. Im sicheren Speicher des HSM besteht ein besonderer Ausleseschutz. Dennoch gibt es Nachteile bezüglich der Performanz, wenn es mehrfach zu einem Zugriff kommt. Im externen Speicher lassen sich

durch den MK sämtliche PKI-Schlüssel verwenden. Nachteilig ist hier, dass alle PKI-Schlüssel auslesbar und nicht weiter wie im HSM durch eine Authentifizierung geschützt sind, falls der MK kompromittiert wird. [44]

In beiden Fällen ist bei der Verwendung eines Master Key darauf zu achten, dass es bei Verlust oder Beschädigung des HSM-Moduls auch zum Verlust der gespeicherten PKI-Keys kommt. Dadurch, dass in diesem Fall das HSM nicht mehr auslesbar ist, ist der Master Key nicht mehr vorhanden und der verschlüsselte Datensatz im externen Speicher auch nicht mehr zu entschlüsseln. Das Handling eines Master Key wird im HSM meistens über eine physikalisch nicht klonbare Funktion – PUF (physical unclonable Function) – betrieben, die im folgenden Abschnitt erläutert ist.

#### 2.2.2.1 Physikalisch nicht klonbare Funktionen – PUF

Eine physikalisch nicht klonbare Funktion – PUF (physical unclonable Function) - ist nach Definition von Müller et al. [45] eine verrauschte Funktion, die in einem physikalischen Objekt respektive einer integrierten Schaltung enthalten ist. [46] Die Antwort $r$ einer Herausforderung $ch$ ist somit abhängig von den gerätespezifischen, physikalischen Eigenschaften der PUF, die in Gleichung 2.1 dargestellt ist.

$$r = PUF(ch) \tag{2.1}$$

Im Kern sind PUFs einem Rauschen ausgesetzt, das durch Umgebungsschwankungen, wie Spannung und Temperatur, zu unterschiedlichen Reaktionen bei gleichen Herausforderungen führt. Die Anwendungsmöglichkeiten beziehen sich z. B. auf die Speicherung eines irreversiblen Elementes

(Fuse-Register), wie einen individuellen Schlüssel oder eine eindeutige ID. Auch werden sie für Authentifizierungsverfahren verwendet.

Nach [45] gelten PUFs als physikalisch nicht klonbar sowie unvorhersehbar und werden als zuverlässig angesehen. Deshalb besteht ein hoher Grad an Manipulationssicherheit, da selbst invasive Eingriffe am Halbleiter im Rahmen der Fertigung eine Reaktion hervorrufen.

So kann eine Hardware zum Schutz der Geheimhaltung eines privaten Schlüssels verwendet werden. Ein über eine PUF-Funktion abgedeckter Zufallszahlengenerator (TRNG) kann ein Datum generieren und im sicheren, nichtflüchtigen Speicher ablegen. Anhand dieser Zufallszahl wird nun im Prozessor ein öffentliches/privates Schlüsselpaar generiert und weitere Signierungsmöglichkeiten werden bereitgestellt. [47, 48]

Ein weiteres Szenario beinhaltet die Verwendung und Erzeugung von symmetrischen Schlüsseln. So können Daten verschiedener Klienten innerhalb eines Gerätes über eine Verschlüsselung gemeinsam genutzt werden. [16] Zudem werden PUFs als Generatoren für symmetrische Schlüssel in Produkten wie der Xilinx UltraScale Zynq FPGA [49] oder Altera-FPGAs [49] verwendet.

## 2.3 Secure Boot, Secure Execution

Die Grundlage einer missbräuchlichen Autorisierung und Verwendung von Steuerungssoftware ist oftmals auf Designfehler in Soft- oder Hardware zurückzuführen. Der Lösungsansatz, um ein System sicher betreiben zu können, liegt in der Bildung einer Vertrauensinstanz.

### 2.3.1 Secure Boot

Eine vertrauenswürdige Architektur basiert auf vertrauenswürdiger Hardware, die über Hardware-Security-Module gebildet wird. Im Zentrum steht ein initialer Prozess zur Gewährleistung der Glaubwürdigkeit der später betriebenen Software. Dieser wird über einen vertrauenswürdigen Bootprozess (Secure Boot) realisiert. Im Bootprozess ist initial die Überprüfung der Software-Integrität, aber auch der gesamten Hardware während des Systemstarts enthalten. Abbildung 2.4 zeigt eine Bootreihenfolge mit einem Bootloader und einer Zielapplikation. Bei jeder neuen Instanz wird durch den gegebenen öffentlichen Schlüssel eine Validierung des zunächst bootenden Codeblocks vorgenommen. [3, 50]

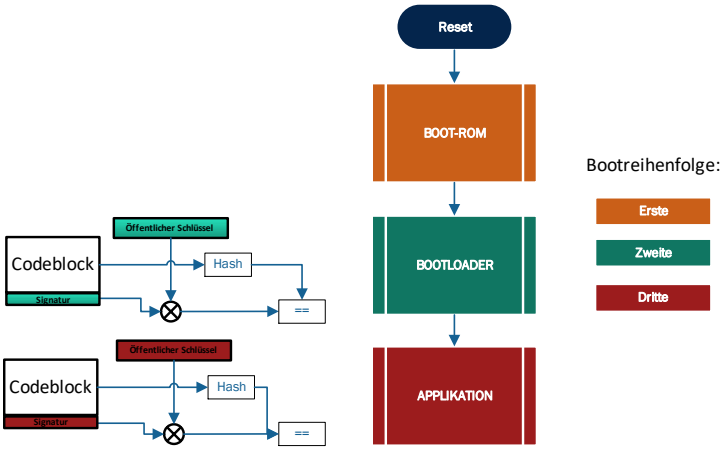


**Abbildung 2.4:** Bootreihenfolge mit Bootloader und Zielapplikation

Beginnend mit der vertrauenswürdigen Quelle wird die nächstfolgende Instanz nach einer Prüfung für vertrauenswürdig erklärt. Beispielsweise besteht die vertrauenswürdige Quelle aus einem gesicherten respektive irreversibel abgelegten Public Key, der wiederum die nächste Softwareinstanz durch Überprüfung von dessen Signatur als vertrauensvoll erklärt. Die Überprüfung läuft ebenfalls über eine vertrauenswürdige Hardware, wie die eines Hardware-Security-Moduls aus Abschnitt 2.2.2. Die Vertrauenskette wird somit nacheinander aufgebaut.

Bei jedem Wechsel einer Bootinstanz werden auch Rechte in Form von Kontroll- und Zugriffsrechten für das System übertragen. Somit muss sichergestellt sein, dass die Kontrollrechte erst übertragen sind, wenn die Integrität gegeben ist. Hier spricht man von einem sequentiellen Validierungsverfahren.

Bootvorgänge weisen oftmals eine enorme Zeitkritikalität auf, wenn es um die zeitliche Bereitstellung der Softwarefunktionalität geht. Neben dem sequentiellen Validierungsverfahren gibt es dahingehend auch das parallele Validierungsverfahren, welche eine vorzeitige Ausführung von Programmcode zulässt.

### 2.3.2 Secure Execution

Bedeutend ist jedoch auch die Absicherung der Ausführung einer Applikation. So kann ein Secure-Boot-Prozess sicherstellen, dass die verwendete Steuerungssoftware validiert ist, er liefert jedoch keinen Schutz, wenn die Funktionalität innerhalb eines Netzwerkes mit anderen Komponenten in Verbindung steht. [51]

Dadurch wird für den Betrieb eine gesicherte Umgebung verwendet. Hier liegt der Kernansatz darin, dass eine Differenzierung von kritischen Funktionalitäten vorgenommen wird. Kritische Prozesse werden in einer sicheren Umgebung ausgeführt und parallel werden Strukturen betrieben, die einen eigenen Adressbereich im Speicher führen, aber auch eigene Peripherien nutzen.

## 2.4 Validierungsmethoden der Softwareintegrität

Im folgendem Abschnitt werden Methoden zur Sicherstellung der Software Integrität nach [43, 21, 41, 1, 52] erläutert. Im wesentlichen bestehen drei Verfahren für die Validierungsmethoden.

### 2.4.1 Hash basiertes Verfahren

Bei umfangreichen Dokumenten oder verwendeter Software kann der Zeitbedarf sehr groß sein, um diese mit einem Integritätsschutz, wie beispielsweise einer Verschlüsselung, versehen zu können. Ein Lösungsansatz ist der Einsatz einer schnell zu berechnenden Prüfsumme (Hashwert). Im Kern wird anhand einer großen Nachricht eine kleinere Zeichenfolge erstellt, die über die Eigenschaft einer Kollisionssicherheit verfügt. Nach Swoboda et al. [1] soll eine gute Hashfunktion kollisionsresistent sein. Nach Watjen [41] spricht man hier bereits von einer kryptografischen Prüfsumme. Die konkrete kryptografische Prüfsumme am Beispiel SHA-1 und SHA-2 ist im Anhang A.1 erläutert. Abbildung 2.7 zeigt das Hash basierte Verfahren.

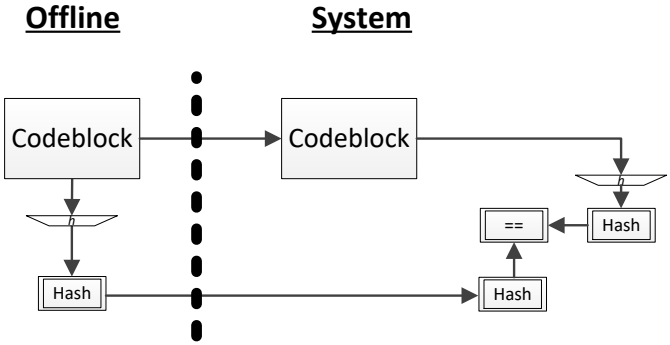


**Abbildung 2.5:** Hash basiertes Verfahren

Der im System zu validierende Codeblock wird im Vorfeld (Offline) einer Hashberechnung unterzogen. Dieser resultierende Hash wird nun im Vertrauensanker (OTP-Register) im System hinterlegt. Im Validierungsprozess wird nun der im System hinterlegte Codeblock einer Hashberechnung unterzogen und der resultierende Hash mit dem im OTP-Register verglichen. Beide Hashwerte müssen zwingend übereinstimmen um die Sofwareintegrität sicherzustellen.

### 2.4.2 MAC basiertes Verfahren

Das Message Authentication Code Verfahren basiert auf der Verwendung eines symmetrischen Schlüssels und kommt häufig bei Transportprotokollen zum Einsatz, wenn es um die Validierung des Integrität einer Botschaft geht. Hinzu auch um die Authentizität bei Kenntnis des symmetrischen

Schlüssels im Rahmen des Nutzerkreises [41]. Eine weiterführende Einleitung zur Berechnung symmetrischer Verschlüsselungsverfahren ist im Anhang im Abschnitt A.2.1 hinterlegt.

Mit Hilfe einer kryptografischen Rechenoperation wird unter Verwendung des symmetrischen Schlüssels eine kryptografische Prüfsumme (MAC) gebildet. Abbildung 2.6 zeigt das MAC basierte Verfahren.

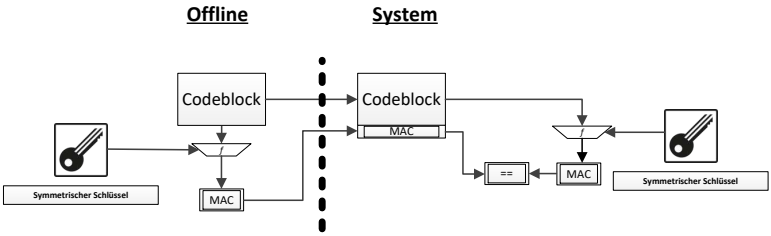


**Abbildung 2.6:** Hash basiertes Verfahren

Der im System zu validierende Codeblock wird im Vorfeld (Offline) mit Hilfe eines symmetrischen Schlüssels einer definierten kryptografischen Berechnung unterzogen. Die dadurch entstandene kryptografische Prüfsumme (MAC) wird systemseitig dem Codeblock als Epilog angefügt.

Im Vertrauensanker (OTP-Register) ist nun der symmetrische Schlüssel hinterlegt. Neben dem irreversiblen Status ist zudem die Voraussetzung geboten, den Schutz vor unbefugten Zugriff und Auslesung des symmetrischen Schlüssels sicherzustellen, da sonst das Geheimnis um dem symmetrischen Schlüssel kompromittiert ist.

Im Validierungsprozess wird nun der im System hinterlegte Codeblock einer mit Hilfe des im OTP hinterlegtem symmetrischen Schlüssels einer definierten kryptografischen Berechnung unterzogen und der resultierende MAC mit dem MAC im Epilog des Codeblockes verglichen. Beide kryptografische Prüfsummen (MAC) müssen zwingend übereinstimmen um die Softwareintegrität sicherzustellen.

### 2.4.3 Signatur basiertes Verfahren

Ein weiterer wichtiger Ansatz ist die Anwendung von elektronischen Signaturen bei Authentizitätsansprüchen für Software, beispielsweise im Desktop-Bereich bei einem Softwareupdate einer Anti-Virus-Software oder bei einem Betriebssystem. Dies gilt auch im Embedded Bereich für die Authentifizierung eines Codeblockes. Hier kommen vermehrt Hardware-Security-Module zum Einsatz, die im Abschnitt 2.2.2 erläutert sind und eine Authentifizierung, Validierung bezüglich des Integritätsschutzes der Software sicherstellen.

Eine konkrete kryptografische Berechnung am Beispiel Rivest-Shamir-Adleman-Verfahren ist im Anhang A.2.2.1 erläutert. Abbildung 2.7 zeigt das Signatur basierte Verfahren.

Der im System zu validierende Codeblock wird im Vorfeld (Offline) einer sogenannten Signierung unterzogen. Unter Zuhilfenahme eines asymmetrischen Schlüsselpaares wird nun der zu signierende Codeblock einer Hashberechnung unterzogen und der resultierende Hash mit dem privaten

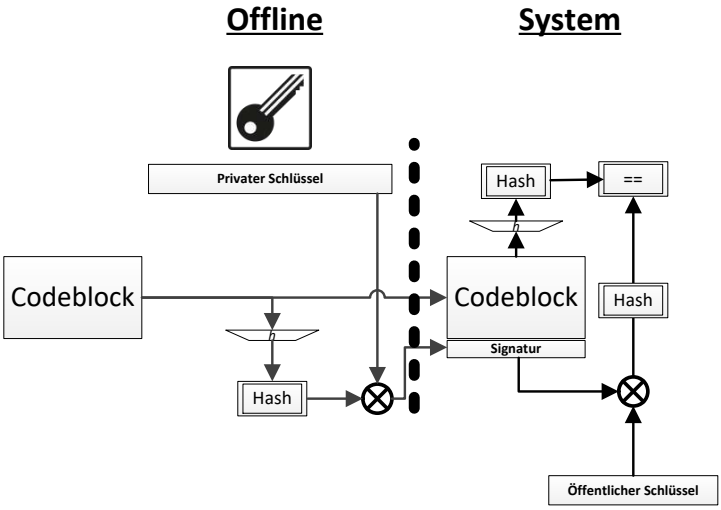


**Abbildung 2.7:** Signatur basiertes Verfahren

Schlüssel verschlüsselt. Es bildet sich somit die Signatur, welche systemseitig als Epilog dem Codeblock angefügt wird.

Der im OTP-Register hinterlegte öffentliche Schlüssel bildet nun den Vertrauensanker. Im Validierungsprozess wird der im System hinterlegte Codeblock einer Hashberechnung unterzogen. Zudem die Signatur mit dem öffentlichen Schlüssel entschlüsselt, womit der im Signierungsprozess entstandene Hashwert entsteht. Beide Hashwerte müssen zwingend übereinstimmen um die Sofwareintegrität sicherzustellen.

## 2.5 Gewählte Taxonomie für Speicherbereiche

Im weiteren Verlauf wird auf die visuelle Darstellung der Arbeit zum Handling von Code-Blöcken sowie den Aufruf des Programmcodes eingegangen. Dazu werden das Memory Mapping und das sichere Aufrufen und Betreiben von Software dargelegt, die insbesondere in den nachfolgenden Kapiteln des Block-orientierten Grundlagenmodelles in Abschnitt 5.1 Anwendung finden.

### 2.5.1 Memory Map

Stetige neue Innovationen führen zu einer zunehmenden Integration zusätzlicher Software-Subsysteme. Die Bewältigung dieser Komplexität gerade hinsichtlich kleiner SoC und deren Hardwareunterstützung sind mit speziellen Bedürfnissen verbunden, insbesondere bei unerlaubten Speicherzugriffen bei Mehrkern-Prozessorarchitekturen [4]. Daher ist es wichtig, im Produktdesign auf die Updatefähigkeit und Einbindung zusätzlicher Subsysteme ohne die Reduzierung der Hauptanwendungen oder Verfügbarkeiten im Speicher zu achten. [51] Eine genaue Definition der im Automobile-Bereich genutzten Speicherarten befindet sich im Abschnitt A.4.

Abbildung 2.8 zeigt das Beispiel eines System-Adressbereiches mit Betrachtung eines Codeblockes. Auf der linken Seite befindet sich der System-Adressbereich. Dieses Mapping schreibt vor, welche Komponenten eines Systems verortet sind. Zusätzlich werden hier in Abhängigkeit der verwendeten SoC-Architektur auch Zusatzkomponenten definiert. Softwarekomponenten, wie z. B. eine Zielapplikation, befinden sich in einer Speicherpartition, die beispielsweise einem externen Speicherbaustein zugeordnet

ist. [5, 53]

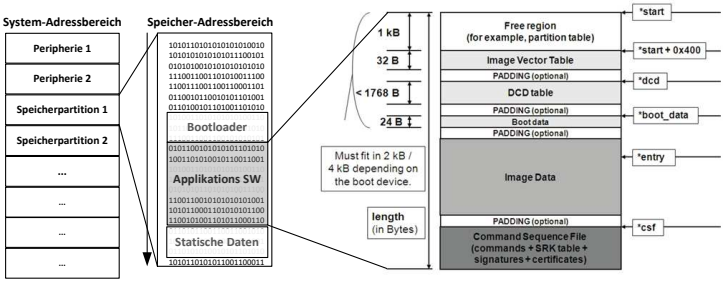


**Abbildung 2.8:** Memory Mapping mit Betrachtung eines Codeblockes nach [4] und [5]

Rechts im Bild ist eine Zielapplikation, die in mehrere Komponenten unterteilt ist. Diese können Metadaten enthalten und werden für Security-Funktionalitäten benötigt. Beispielsweise sind hier in der Image Vector Table Blockgrößen und die Länge des eigentlichen Programmcodes (Image) angegeben. Abschließend werden Komponenten wie Zertifikate angehängt, die eine Überprüfung der Softwareintegrität sicherstellen. [5, 51]

In einem SoC werden mehrere Adressebenen implementiert. Insbesondere trifft dies bei Mehrkernarchitekturen zu. Dahingehend kommt eine Memory Management Unit zum Einsatz, die eine Adressraumtrennung verwendet. Prinzipiell sind nach [4] drei Adressebenen definiert:

- Physische Adresse (PA), die die Adresse im System-Adressbereich repräsentiert.
- Physische Zwischenadresse (IPA), die eine indirekte Ebene einführt (hier am Beispiel Speicher-Adressbereich).

- Virtuelle Adresse, die durch eine Applikation respektive ein Betriebssystem verwaltet wird und losgelöst von der physischen Adresse PA ausgelagert wird.

Dieses Kapitel umfasste die Einleitung und Anwendung wichtiger kryptografischer Verfahren und deren Einsatzmöglichkeiten in einem Steuergerät. Neben relevanten kryptografischen Prozessen zum Signierungs- und Validierungsverfahren wurde in Abschnitt 2.3.1 eine Modellebene eingeführt, die im späteren Verlauf, insbesondere im Block-orientierten Grundlagenmodell in Abschnitt 5.1, Anwendung findet. Weiterführend wird sich im nachfolgenden Kapitel mit dem Lebenszyklus des Steuergerätes selbst beschäftigt.

# 3 Automotive Electronic

Im vorherigen Kapitel wurden kryptografische Methoden und deren Anwendungen in Steuergeräten vorgestellt. Dieses Kapitel betrachtet vertieft den Kerngedanken der Security, angewandt auf elektronische Komponenten im automobilen Bereich AE (Automotive Electronic) und dem gesamten Produktlebenszyklus, damit die Voraussetzungen zur Forschungsfrage im Kapitel 4 geschaffen werden können.

Im Kern wird in den folgenden Abschnitten der Durchlauf eines Produktes durch den gesamten Produktlebenszyklus (PLC) betrachtet und das allgemeine Konzept eines PLC vorgestellt. Separiert in den jeweiligen Phasen sind Grundlagen für Produktanforderungen, die Serienphase und eine fokussierte Betrachtung der Nachserienversorgungsstrategien sowie deren Lifecycle Managements enthalten, um moderne Fertigungsansätze zur Bewältigung der Security-Eigenschaften vorzustellen.

## 3.1 Der Produktlebenszyklus

Dieser Abschnitt stellt den Product Life Cycle (Produktlebenszyklus, PLC) eines Produktes vor. Prinzipiell beschreibt das Modell einen gesamten Durchlauf, beginnend bei der Konzeptionierung, Entwicklung, Fertigung und Nachserienversorgung bis zum Auslauf und der Abkündigung des Produktes. Der Produktlebenszyklus ist nach Wirtz [6] in Abbildung 3.1

unter zwei Modellen zusammengefasst und betrachtet das Produkt sowie dessen Markt, in dem es mit einem entsprechenden Volumen vertrieben wird.

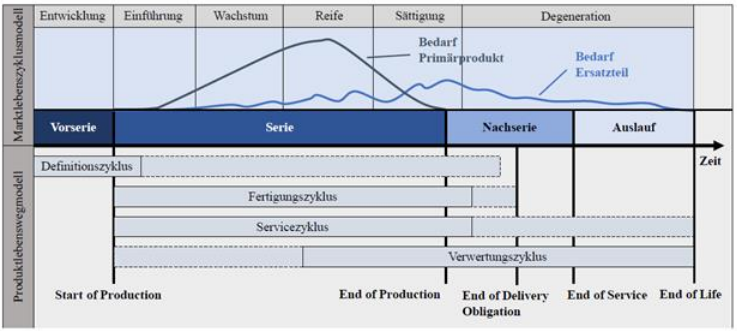


**Abbildung 3.1:** Modell des Produktlebenszyklus nach Wirtz [6]

Der PLC wird auch als Marktlebenszyklusmodell bezeichnet und gibt die vier Phasen eines Lebenszyklus vor, die hier farbig dargestellt sind. Dieses Modell besteht aus einer Vorserie, der eigentlichen Serienproduktion, der Nachserie und dem Auslauf. Diese Phasen lassen sich nochmals unterteilen, die oberhalb der Voluminakurve angegeben sind. In Rahmen der Vorserie steht die Entwicklung und der Produktentstehungsprozess. Die Serienphase enthält die Einführung (Ramp-Up), das Wachstum (Produktionssteigerung), die Reife (stabile Serienproduktion), die Sättigung (Produktionsminimierung) und die Degeneration. In dieser letzten Phase läuft das Produkt am Markt aus. Die Voluminakurve beschreibt einmal

den Bedarf des Primärproduktes (Endproduktes) und dessen Bedarf an Teilkomponenten im Sinne von Ersatzteilen. [54, 35]

In Analogie zu den vier Hauptphasen (Vorserie, Serie, Nachserie und Auslauf) definiert im unteren Teil des Bildes das Produktlebenswegmodell ebenfalls vier Zyklen. Diese orientieren sich an den Meilensteinen einer Serienfertigung. Mit der Beendigung der ersten Phase (Vorserie) beginnt der Produktionsstart (Start of Production – SOP). Die Serie dieser Phase endet mit dem End of Production (EOP), womit die Phase der Nachserie gestartet wird. Innerhalb dieser Phase befindet sich der Meilenstein End of Delivery Obligation, wo es zu der Abkündigung des Produktes kommt. Ist die Phase der Nachserie beendet, wird auch der Meilenstein End of Service (EOS) erreicht. Anschließend kommt die Auslaufphase, die mit dem End of Life (EOL) abgeschlossen wird. [54]

Aus Sicht der Zyklen überlappt sich der Definitionszyklus mit der Vorserie und der Entwicklung. Tatsächlich kann der Definitionszyklus in andere Phasen des PLC ausgedehnt werden, da es in diesen auch zu Änderungen am Produkt kommen kann bzw. Weiterentwicklungen notwendig sein können. Der Fertigungszyklus beginnt mit der Serienfertigung zum SOP und geht bis in die Nachserie hinein. Ersatzteilbedarfe werden durch die geringe Menge anfangs aus der Serie entnommen. Im späteren Verlauf zum EOD wird ein Last Call gemacht, wodurch der OEM das Volumen eines finalen Fertigungsloses bestimmt. Der Servicezyklus ist eine Betreuung des Kunden hinsichtlich bestehender oder benötigter Produktionsvolumina für Ersatzteilbedarfe. Der abschließende Verwertungszyklus behandelt gebrauchte Primärprodukte (oder auch Ersatzteile), die im Rahmen der Nachserien-Versorgungsstrategie gehandelt werden. [10, 54, 35]

In den nachfolgenden Abschnitten werden die einzelnen PLC-Phasen genauer betrachtet.

## 3.2 Vorserie (Phase I)

Die erste Phase im PLC beschreibt im Kern den entwicklungstechnischen Ansatz, bis im Rahmen des Produktentstehungsprozesses die Vorserie aufgenommen wird.

Für die Produktentwicklung stehen dabei die Produktanforderungen und -haftung im Vordergrund. Hierbei wird insbesondere für Zulassungsverordnungen der Safety-Aspekte die Automobilfunktionssicherheit nach ISO 26262 beschrieben. Weiterführende Security-Aspekte werden nach ISO 21434 und SAE J3061, auch hinsichtlich wichtiger Anforderungen des Hardwaresupports, der Sicherheitsintegration und Firmware-Updates, betrachtet.

Anschließend wird auf den Softwareentstehungsprozess eingegangen. Mit der Entwicklung von Software für den automotiven Bereich werden auf zusätzliche Teststrategien hingewiesen. Weiterführende Informationen zu Referenzarchitekturen werden im Anhang unter Abschnitt A.5 am Beispiel der AUTOSAR-Plattform vorgestellt.

Final wird auch auf die Plattformentwicklung eingegangen, wobei deren Herausforderungen im automobilen Kontext dargestellt werden.

### 3.2.1 Produktanforderungen

Beginnend mit der Ausschreibung eines Projektes für ein neues Produkt müssen viele Randbedingungen beachtet werden. Produktanforderungen werden im Kern durch drei Bereiche beeinflusst, die in Abbildung 3.2 dargestellt werden.

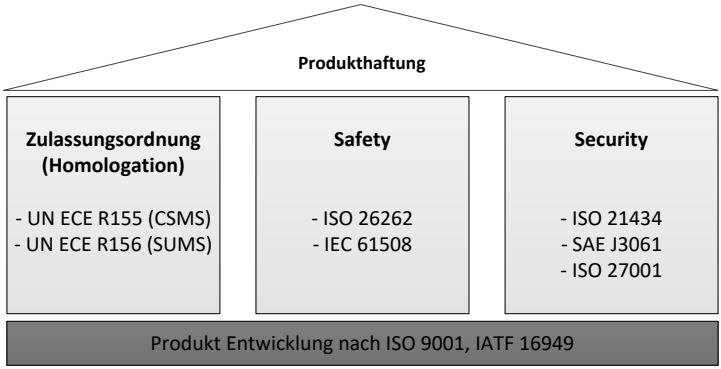


**Abbildung 3.2:** Säulen der Produkthaftung nach [7]

Basierend auf der Produktentwicklung nach ISO 9001 und IATF 16949 (ehemals ISO TS) [55] stützen sich darauf aufbauend die Bereiche der Zulassungsordnung, der Safety-Aspekte und Security-Anforderungen.

#### 3.2.1.1 Zulassungsordnung

Neben der EU-Zulassungsverordnung zur Schaffung eines Rahmens für die Genehmigung von Kraftfahrzeugen [56] gibt es ebenfalls seit 2020 vom UNECE World Forum for Harmonization of Vehicle Regulations (UN ECE WP.29) neu erstellte respektive verabschiedete Regelwerke. Diese Regelwerke sollen einen Rahmen insbesondere für die Cybersicherheit von Fahrzeugen setzen.

Somit behandelt das Regelwerk UN ECE R155 [57] ein zertifiziertes Cybersecurity-Managementsystem (CSMS). Dadurch soll bereits im Produktentstehungsprozess die Abischerung von Fahrzeugen behandelt werden, um eine Risikominimierung von Cyberangriffen herbeizuführen.

Das Regelwerk UN ECE R156 [58] schreibt als künftige Bedingung für die Typgenehmigung ein Softwareupdate-Managementsystem (SUMS) vor, das insbesondere auf das Management Flotten-übergreifender Softwareupdates over the Air (FOTA) abzielt. Insbesondere sollen so auch im Kontext des Flottenschutzes Vorfälle oder Sicherheitslücken über Fahrzeugflotten hinaus erkannt werden. [7]

#### 3.2.1.2 Safety-Aspekte

Mit der Einführung komplexer Technologien steigt die Herausforderung für die Produktentwicklung. Somit sind Automatisierungsansätze und Standardisierungen der funktionalen Sicherheit im Produktdesign unerlässlich, um Konstruktionsaktivitäten zu unterstützen. Hierbei trägt die ISO 26262 [59] eine maßgebliche Rolle. [60, 61]

Als eine Weiterentwicklung der IEC 61508 wurde 2011 die ISO 26262 veröffentlicht. Für die Gültigkeit elektronischer Fahrzeugsysteme bis zu

3,5 Tonnen ist dieser Industriestandard von vielen Fahrzeugherstellern anerkannt. Dies schließt auch sämtliche Lieferanten der gesamten Lieferungskette mit ein. Durch die Einhaltung soll das Haftungsrisiko im Fehlerfall, wie bei einem Ausfall von Fahrerassistenzsystemen, reduziert werden. Somit ist in jeder Phase einer Produktentwicklung sichergestellt, dass die ISO 26262 eingehalten wird.

Verschiedene Sicherheitsanforderungen decken Bereiche funktionaler Sicherheit ab, die während der gesamten Lebensdauer fehlerhaft auftreten können. Im Kern geht es um die Wahrscheinlichkeit eines Fehlerfalles, basierend auf dem Gefahrenpotential, der Kontrollierbarkeit und der Unfallschwere. So steht Severity $S$ für die Schwere des Fehlers, der eine Gefährdung des Nutzers oder der Umgebung beschreibt. Exposure $E$ ist die Eintrittswahrscheinlichkeit, somit die Häufigkeit oder Dauer eines Betriebszustandes. Controllability $C$ bezeichnet die Beherrschbarkeit des Fehlers. Diese Bereiche unterteilen sich jeweils aufsteigend nach der Intensität in vier Level. [8, 62]

Aus den aufgeführten Komponenten ergibt sich die Schlüsselkomponente des Standards, der ASIL-Einstufung (Automotive Safety Integrity Level – ASIL), in der Risiken für den Fahrer und andere Verkehrsteilnehmer im Störungsfall bewertet werden. Die Bewertung fokussiert sich ausschließlich auf den Personenschaden und nicht auf die in den Systemen verwendeten Technologien und Bauteile. Die Level ergeben sich nach der ISO 26262 aus den Sicherheitsleveln anhand eines Punktesystems, deren Verteilung in Abbildung 3.3 dargestellt ist. Zu beachten ist, dass der Sprung zwischen ASIL Stufe B und C besonders groß ist. Hier kommen anhand der Zuverlässigkeitsanforderungen Verpflichtungen an die eingesetzte Architektur zum Tragen, beispielsweise Redundanz oder zweikanalige Lösungen. [8]

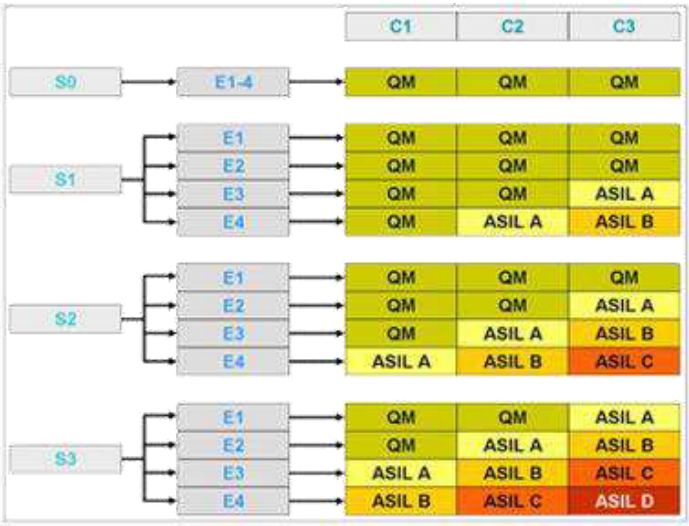


**Abbildung 3.3:** Verteilung des Punktesystems anhand der Sicherheitslevel nach ISO 26262 aus Vogt [8]

Bei der Entwicklung resultieren aus Produktdaten wichtige Hilfestellungen, um die Klassifizierungen einordnen zu können. Weitere Informationen zu Rückläufern und auftretenden Problemen spielen ebenfalls eine wesentliche Rolle. [60, 61] Für die Produktentwicklung ist es dafür auch notwendig, eine Fehlermöglichkeiten-Einflussanalyse (FMEA) zu erstellen. Eine Erläuterung nach Hering und Schloske in [63] befindet sich im Anhang im Abschnitt A.5.

#### 3.2.1.3 Security-Aspekte

Die Einhaltung von sicherheitsrelevanten Funktionen schließt auch deren Sicherheit im Sinne von Security mit ein. Die ISO/SAE 21434 [64] beschreibt im Kern eine Risikoidentifikation. Zielsetzung ist es, ein Sicherheitskonzept zu erstellen, das auf Basis vorab identifizierter Risiken aufbaut. Insbesondere im vollführten Sicherheitskonzept in Form der Bedrohungs- und Risikoanalyse (Threat and Risk T&R-Analyse) im Kapitel 6 und der durchgeführten Risikoanalyse im Abschnitt 6.6 wird die Risikoidentifikation, auch unter Zuhilfenahme der SAE J3061 (Automotive Security Requirement Engineering) [65] und in Anlehnung an die ISO 27001 [66], für die Beurteilung und Behandlung von Informationssicherheitsrisiken verwendet.

Nach Zhang et al. [67] liegt eine derzeitige wesentliche Herausforderung im digitalen Transformationsprozess. Moderne Fahrzeuge verfügen über komplexere Netzwerktopologien mit diversen Funkschnittstellen (Bluetooth, Wi-Fi, DSRC, 3G/5G). Diese Schnittstellen werden zur Fahrzeugdiagnose, aber auch als Verbindung zu anderen Consumer-Geräten und Backends für Servicedienstleistungen verwendet. Diese Konnektivität birgt das Potential, zusätzliche Funktionalitäten für Komfortfunktionen wie auch den Transport selbst signifikant zu verändern, [67] beispielsweise durch Car-to-X-Kommunikation und das autonome Fahren. Diese Herausforderungen führen auch zu einer Reihe von Sicherheits- und Datenschutzproblemen. Sicherheitskritisch betrachtet, im Sinne der Funktionalität des Fahrzeugs, werden die im Fahrzeug verbauten Steuergeräte durch den Vernetzungsgrad zunehmend anfälliger für Sicherheitsangriffe. Diese neuen Angriffsvektoren schließen auch Diebstahlsicherungen, Wegfahrsperren und schlüssellosen Fernzugriff ein. Weiterführende Literatur findet sich unter [68], [69],

[70] und [71]. Im Kern sollen nach Zhang et al. [67] die Manipulation und das Abgreifen wichtiger Dateninformationen verhindert werden.

Mundhenk et al. [9] schlagen für präventive, aber auch repressive Maßnahmen ein Angriffserkennungssystem namens „Instrusion Detection System“ (IDS) vor. Diese Methodik ist im Desktop-Bereich bei Anti-Viren-Software bereits lange in Anwendung. Grundsätzlich soll nun auch im Automobilbereich ein ständiger Vergleich des Fahrzeugnetzverkehrs mit einem Grundlinienmodell erfolgen, um Anomalien und Ausfallerscheinungen detektieren zu können. Auch wird die Interaktion mit dem Fahrer untersucht. Jedoch kann in einem sicherheitskritischen Kontext nicht jedes Angriffserkennungssystem automatisiert werden, insbesondere wenn ein Angreifer versucht, den Eingriff verschleiernd zu gestalten. Hierzu gibt es andere Ansätze, wie etwa das Erkennen eines vordefinierten Angriffsmusters auf Basis der Entropie des Datenverkehrs. [9, 72]

Im folgenden Abschnitt werden Aspekte von Security-Anwendungen vorgestellt. Diese befassen sich mit der Verschlüsselung von Daten und dem Hardwaresupport eines Steuergerätes, dessen Sicherheitsintegration im Fahrzeug und Updatemöglichkeiten für die Firmware.

#### 3.2.1.4 Verschlüsselung und Hardwaresupport

Um in Fahrzeugnetzwerken unter Echtzeitanforderungen eine Verschlüsselung umzusetzen, hat die Herstellerinitiative Software (HIS) einen kryptografischen Beschleuniger, „Secure Hardware Extension“ (SHE) genannt,

spezifiziert. Diese Standardisierung soll die Schlüsselspeicherung und Verschlüsselung bei Prozessorarchitekturen unterstützen. Hardware-Security-Module, wie im Abschnitt 2.2.2 beschrieben, intensivieren diese Standardisierung. [9]

Im Wesentlichen geht es bei der Standardisierung darum, Auswirkungen der Latenz bei Echtzeitkommunikation zu begrenzen. Auch sind durch bekannte Angriffsvektoren die Problematiken einer geschützten Kommunikation evident, wenn Schlüssel für eine symmetrische Verschlüsselung häufig in Steuergeräten vorprogrammiert sind. Diese Gefahren haben Auswirkungen auf die Lebensdauer von Fahrzeugen oder möglicherweise kompletten Fahrzeugflotten. Um diesen Skalierungseffekten entgegentreten zu können, gibt es beispielsweise nach [9] Ansätze zur Minderung durch Verschleierung von CAN-Nachrichten-IDs.

#### 3.2.1.5 Sicherheitsintegration

Wie bei jeder Sicherheitsmaßnahme ist die in Abschnitt 2.1 definierte Integration ein wichtiger Garantiefaktor. Im Kern geht es um die Validierung digitaler Zertifikate aus Abschnitt A.3. Die Sicherheit und Authentizität von Zertifikaten hängen vom zugehörigen privaten Schlüssel ab, da dieser den Bestandteil der Attribute im Zertifikat ändern kann. Bei Verlust, Entwendung oder Manipulation eines privaten Schlüssels ist das entsprechende Zertifikat nicht mehr vertrauenswürdig. Es kann somit zum Identitätsklau beispielsweise einer Software, eines Steuergerätes oder von Zugangsberechtigungen kommen, insbesondere bei physischem Zugang eines Steuergerätes. Daher ist, wenn ein privater Schlüssel gestohlen oder absichtlich verteilt wird, das entsprechende Zertifikat nicht mehr vertrauenswürdig.

Wird zum Beispiel der private Schlüssel eines Steuergerätes gestohlen, kann ein böswilliges Gerät dessen Identität annehmen.

Mundhenk et al. beschreiben in [9] das Konzept LASAN (Lightweight Authentication and Authorization). Der wesentliche Aspekt ist ein Authentifizierungsframework zur Gewährleistung der vollen Fahrzeugfunktionalität. Abbildung 3.4 zeigt eine verallgemeinerte Konzeptionierung zur Authentifizierung und Autorisierung von Steuergeräten und Software zueinander. Der obere Bereich stellt eine Fertigungsumgebung dar. Der OEM verwaltet eine Liste gültiger Zertifikate, beispielsweise zur Verifizierung signierter Softwarecontainer, die während der Fertigung in dem Steuergerät installiert werden. Diese Zertifikate werden der Fertigung zur Verfügung gestellt. Alle Datenübertragungen erfolgen in einem gesicherten Netzwerk. In der Produktion wird nun lokal im Steuergerät die Liste der gültigen Zertifikate in einem Sicherheitsmodul (HSM) gespeichert. Parallel stellt eine Zertifizierungsstelle Zertifikate bereit, die ebenfalls der Fertigung übergeben werden. Diese Zertifikate werden jedoch für Authentifizierungen und für Berechtigungen innerhalb einer laufenden Applikation des Steuergerätes verwendet. So kann sich jemand von extern, beispielsweise ein Diagnosetester einer Servicewerkstatt, über die OBD-Schnittstelle authentifizieren und somit Diagnosefunktionen am Steuergerät ausführen. Das Security-Modul prüft hierbei, ob das verwendete Zertifikat gültig ist. Dieses Handling erlaubt zudem auch gegenseitige Authentifizierungen der Steuergeräte.

Für die Aktualisierung einer im Steuergerät befindlichen gültigen Liste erlaubter Zertifikate bedarf es einer Verbindung zum Konfigurationsmanagement des OEM. Hier bilden sich zusätzliche Angriffsvektoren,

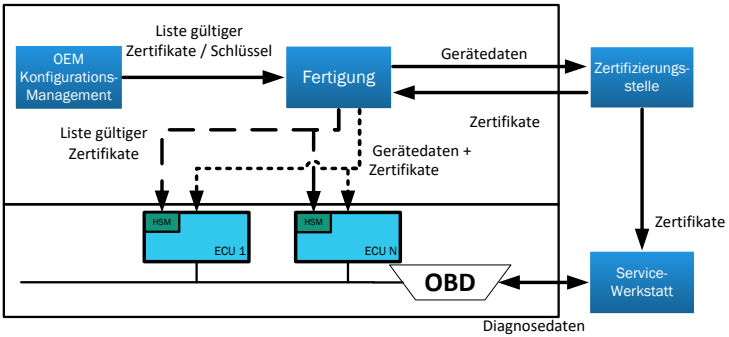


**Abbildung 3.4:** Konzeptionierung zur Authentifizierung und Autorisierung von Steuergeräten und Software zueinander nach Mundhenk [9]

wie etwa Denial-of-Service-Attacken (DoS). Daher ist eine Steuergeräte-Authentifizierung unumgänglich.

#### 3.2.1.6 Firmware-Updates

Die Steuergeräte-Authentifizierung spielt auch bei Softwareupdates, insbesondere bei vernetzten Fahrzeugen, eine wesentliche Rolle. Updates der Art „over the Air“ (OTA) eröffnen die Möglichkeit, Software zu jedem Zeitpunkt zu aktualisieren und somit Ausfallzeiten für einen Besuch in einer Servicewerkstatt zu reduzieren. Hinzu kommt auch noch der Anspruch auf die Integrität und Authentizität der neu einzusetzenden Software. Somit ist auch bei einem Update nicht nur eine Authentifizierung unumgänglich, sondern auch die Überprüfung der einzusetzenden, signierten Software selbst.

Allgemein besteht aus der Security-Perspektive der Automobilindustrie ein zunehmender Entwicklungsbedarf in Bezug auf die im Systembereich verwendeten Informationen, bei denen die Evaluationskriterien des ISO/IEC 15408 (Informationstechnik[IT]-Sicherheitsverfahren für IT-Sicherheit) eine Schlüsselrolle spielen. Hierzu beschreiben die ISO/IEC 18045 die Methoden zur IT-Security-Evaluierung [73], die auch in der Risikoanalyse im Abschnitt 6.6 zum Tragen kommen.

### 3.2.2 Softwareentstehungsprozess

Die neuesten Features eines Fahrzeugs basieren auf softwareintensiven Systemen. Für die Entwicklung, insbesondere den Softwareentstehungsprozess, erfordert dies eine umfangreiche Planung. Das Design muss den Security- und Safety-Anforderungen entsprechen, was eine sorgfältige Planung und Implementierung voraussetzt. Final stehen auch die Überprüfung und Validierung einer Software im Fokus, ehe sie freigegeben werden kann. Oftmals müssen hier tiefgreifende Analysen der eingesetzten Hardware und Abnahmen wie vom Kraftfahrtbundesamt stattfinden.

Staron et al. beschreiben in [74] anhand eines V-Modells einen Softwareentstehungsprozess für die Automobilindustrie.

Die linke Seite des V-Modells beginnt mit den Produktanforderungen. Im Requirements Engineering wird das Design auf funktionaler und Systemebene abgeleitet. In der darin enthaltenen Phase, in der die Systemanalyse durchgeführt wird, werden die Funktionalitäten den Anforderungen zugeordnet. Es folgt anschließend die Erstellung der Softwarearchitektur bis hin zur Implementierung. Hier wird das Design in Quellcodes umgesetzt.

Die rechte Seite des V-Modells beschreibt neben der Implementierung im Kern das Testen und Evaluieren der geschriebenen Software. Implementieren, Testen und Evaluieren stehen in unmittelbarem Zusammenhang mit der iterativen Softwareentwicklung. Zunächst werden Einheits- und Komponententests durchgeführt, die auf gleicher Ebene mit dem Design abgeglichen werden. Anschließend erfolgen weitergehende Funktionsprüfungen und Abnahmen durch den Kunden.

Neben den Integrationsstufen der Softwareentwicklung wird auch das Variantenmanagement betrachtet. Hinzu werden im Folgenden verschiedene Teststrategien und die verwendeten Methoden für diese vorgestellt.

#### 3.2.2.1 Automotive Softwareentwicklung

In der Entwicklung von Software für Steuergeräte herrschen oftmals komplexe Kunden-Lieferanten-Beziehungen. Der OEM könnte beispielsweise einen Zulieferer (TIER1) eine Fahrzeugkomponente anhand eines Lastenheftes anfertigen lassen, jedoch die Softwareentwicklung für die Applikation selbst übernehmen. Auch könnte eine dritte Partei diesen Auftrag erhalten. Im Kern bestehen die Entwicklungsprozesse aus Phasen und Aktivitäten. So entstehen verschiedene Akteure. Diese stellen sich u. a. aus Softwarekonstrukteuren, Softwarearchitekten, Projektmanagern und Qualitätsmanagern zusammen. Prinzipiell sind die Entwicklungsprozesse in Phasen, analog zum V-Modell mit dem Schwerpunkt auf der Softwareentwicklung strukturiert. Das Requirements Engineering befasst sich mit der Technik und dem Tooling zur Erfüllung der Anforderungen. Dies impliziert Methoden zur Ermittlung von Spezifikationen, Dokumentationen, Qualitätssicherungen und eine Priorisierung der Anforderungen. [74, 75, 76]

Nach Staron [74] liegt die Quelle sämtlicher Innovationen in der Software. Softwarefunktionalitäten etablieren sich schneller als einzelne Innovationen. Daher werden die SW-basierten Erneuerungen als systematische Innovation bezeichnet. Diese Eigenschaft spiegelt sich in den Softwareentwicklungsprozessen wider, die hohe und vielseitige Anforderungsspezifikationen bereitstellen.

Zum Softwareentstehungsprozess werden Schichten-orientierte Referenzarchitekturen benötigt. AUTOSAR definiert eine Methodik für die Entwicklung von automobilen Softwaresystemen und liefert die Sprache (Metamodell) für ihre Architekturmodelle. Weiterführende Informationen sind im Abschnitt A.5 und in [25], [77] sowie [78] verortet.

Die Integrationsstufen der Softwareentwicklung gliedern sich in drei Bereiche. Durch die Beteiligung verschiedener Parteien und Komponentenhersteller bildet die erste Stufe die Softwareintegration. Verschiedene Komponenten werden zu einer Gesamtapplikation zusammengefügt und getestet. Die zweite Stufe ist die Eingliederung der Applikation in eine hardwarenahe Umgebung. Zunächst werden üblicherweise Teile der Hardware simuliert, bevor Tests an Evaluations- oder Mustersteuergeräten durchgeführt werden. Das letzte Level ist die Hardwareintegration. Hier wird die im Steuergerät befindliche Software im Verbund und im eingebauten Zustand des Zielsystems getestet. Weiterführende Informationen finden sich unter [75], [79], [80] und [81]

#### 3.2.2.2 Konfigurations-/Variantenmanagement

Der Endkunde hat viele Konfigurationsangebote beim Neukauf eines Autos. Etliche Features können vorab bestimmt werden. Dies geschieht aufgrund der Tatsache, dass sämtliche Konfigurationen softwareseitig variabel bereitgestellt sind. Hier ist das Softwaremanagement entscheidend für die Realisierung der Varianz. Grundsätzlich unterscheidet man zwei Variabilitätsmechanismen.

Der erste Mechanismus ist die Konfiguration. Konfigurationsdaten werden als Parameter in einen statischen Datensatz gelegt. Die Applikation greift fest auf diese Parameter zu und konfiguriert die Verfügbarkeit der bereitgestellten Funktionen. Somit können in einem Softwarestand viele Funktionalitäten abgedeckt werden. Für die Softwareentwicklung stellt dies jedoch einen Mehraufwand dar.

Der zweite Mechanismus ist eine feste Programmierung der jeweiligen Varianz. Hier liegt die Herausforderung im Softwaremanagement bei dem Handling vielseitiger Softwareversionierungen. Auch spielt die Wiederverwendung bereits existierender Projekte/Software eine wesentliche Rolle. In der Regel stellt ein neues Projekt eine Iteration des vorherigen dar. [82, 83]

#### 3.2.2.3 Teststrategien

Teststrategien arbeiten in Gegenlauf zum Requirements Engineering. Hauptsächlich wird mit einem Basistest in Form eines Unit-Tests begonnen. Der Unit-Test prüft jede Funktion und Codezeile im Programm. Kernziel ist es, Mängel im Zusammenhang mit der Implementierung zu finden. Solch

ein Unit-Test wird automatisiert durchgeführt, wobei benötigte Daten und Variablen generiert bzw. bereitgestellt werden. [67]

In den darauffolgenden Testphasen werden ganze Komponenten und verbundene Einheiten im Rahmen eines Integrationstests untersucht. Hier liegt der Fokus auf der Prozesskommunikation von Schnittstellen einzelner Komponenten, die funktional miteinander verbunden sind. Auch wird es hinsichtlich möglicher Speicherverletzungen bei falscher Auslegung eines System- und Speicher-Adressbereiches (siehe 2.5.1) geprüft. Im Automotive-Bereich werden diese Tests in einer Simulation der Umgebung durchgeführt.

Anschließend folgen Systemtests, in denen das System mit den gestellten Anforderungen seine Spezifikationen erfüllt. Hierbei liegt der Fokus auf mehreren Aspekten. Die Funktionalität muss den Anforderungsspezifikationen entsprechen. Die Interoperabilität prüft die Interaktion verschiedener Systeme. Zudem werden Leistungs- und Belastungstests durchgeführt, um eine Skalierbarkeit beschreiben und abgleichen zu können. Dazu liegt der Fokus auch auf der Zuverlässigkeit. Sämtliche Aspekte werden zusätzlich mit Anforderungen und behördlichen Vorschriften abgeglichen. [76, 84]

#### 3.2.2.4 Auswertung

Das Design einer entwickelten Architektur ist nach Staron [74] oft an Prinzipien orientiert. Softwarearchitekten verfolgen häufig einen Stil, der bis in die Verteilung von Signalverläufen in Kommunikationsbussen reicht. Die Norm der ISO/IEC-25000-Reihe beschreibt Softwarequalitätsanforderungen und -bewertungen und gibt eine Hilfestellung für die Kernfrage

der messbaren Verbesserung oder gar Verschlechterung einer Architektur. [74] Die Weiterentwicklung der Automobilsoftware setzt auch einen Anspruch auf fortgeschrittene Softwareentwicklungsmethoden voraus. Durch die zunehmende Menge der Komplexität an Software ist auch vermehrt ein Fokus auf die Rückverfolgbarkeit und Prozesse für die Verfolgung von Anforderungen nach der ISO 26262 gerichtet. Auch gewinnen nichtfunktionale Eigenschaften wie das Verhalten bei erhöhter Kapazität von Kommunikationsbussen sowie deren Synchronisation und Überprüfung an Bedeutung.

Schon in den ersten validierten Softwareversionen werden Prototypen erstellt. Zunächst werden mit erweiterten Hardwarekonfigurationen bereits entsprechende Muster geschaffen, die einen Freigabeprozess durchlaufen.

### 3.2.3 Plattformentwicklung

Die im Abschnitt 2.2.0.1 vorgestellte Zentralisierung von Steuergeräten und der darin enthaltenen Transformation von Domänen-orientierten hin zu Zonen-orientierten Netzwerktopologien übt auch Einfluss auf zentralisierte und modularisierte Aufbauten von Hardwarearchitekturen aus. So wurden anstelle von dedizierten Hardwarekomponenten zentralisierte Controller/Mehrzweckprozessoren eingesetzt.

Grund für diese Entwicklung sind die Verfügbarkeiten der Bauteile sowie der zunehmende Kostendruck und die Gewährleistung und Zuverlässigkeit des Gesamtsystems. Auch die Leistungsfähigkeit spielt eine wesentliche Rolle. Produktentwicklungszyklen werden durch rasante Entwicklungen in der Kommunikationstechnik und den Trend der Digitalisierung enorm verkürzt. Um im globalen Wettbewerb Schritt halten zu können, muss eine

Anpassung der Nachfrage erfolgen. Diese Anpassung spiegelt sich nicht nur in verringerten Losgrößen, sondern auch in der damit verbundenen Erhöhung der Produktionskosten wider.

Auch hinsichtlich des automobilen Bereichs ist dies eine enorme Herausforderung, da lange Nachserienversorgungsverpflichtungen eingehalten werden müssen. Mit dieser aufkommenden, zunehmenden Produktvielfalt müssen agile Methoden eingesetzt werden, um die Herausforderungen für innovativere und komplexere Produkte bewältigen zu können. Daher ist es wichtig, auf einheitlichen Plattformen mit hoher Wirksamkeit und Effizienz die Komplexität zu beherrschen. [85, 86]

## 3.3 Serienphase (Phase II)

Nachdem die Entwicklungsphase (Phase I des PLC) abgeschlossen ist und auch die ersten Prototypen erfolgreich abgenommen wurden, beginnt die Serienphase. Eine Serienphase definiert sich aus dem Zeitraum zwischen dem Beginn (SOP – Start of Production) und dem Ende der Serienfertigung (EOP – End of Production). Nach EOP werden Nachserienversorgungsstrategien aufgrund von Lieferverpflichtungen und Versorgungsgarantien angewendet. Diese Phase wird mit einem angekündigten Lieferungsstopp (EOD – End of Delivery Obligation) und dem Auslauf der Nachserienverpflichtung (EOS – End of Service) beendet. [10, 11]

Innerhalb der Serienproduktion werden Stückzahlen in Abhängigkeit von einem Abruf hergestellt. Die Produktionsphasen zwischen SOP und EOP gliedern sich in vier Bereiche, die in Abbildung 3.5 dargestellt sind.

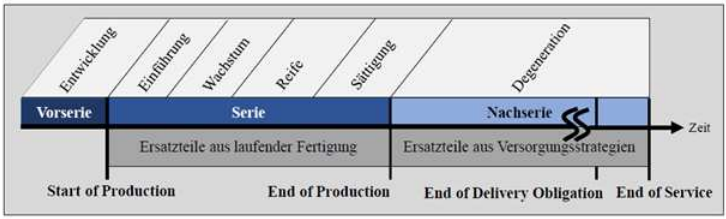


**Abbildung 3.5:** Phasen der Ersatzteilversorgung aus [10] nach Bothe [11]

Eine Serienproduktion beginnt mit einer Einführungs- und Wachstumsphase. Diese beiden Phasen werden als Hochlaufphase bezeichnet und der Phase zwei des PLC zugeordnet. Abschnitt 3.3.1 beschreibt die Hochlaufphase mit den strategischen Eigenschaften und Zielgrößen. Es schließt sich eine Erläuterung über die Herausforderungen der Reife und Sättigung einer Serienproduktion an, die zusammenfassend als Serienphase bezeichnet wird und der Phase drei des PLC zugeordnet ist.

### 3.3.1 Hochlaufphase

Eine Hochlaufphase definiert sich durch die Anschaffung und Planung einer Serienfertigung und deren Equipment. Charakteristisch sind die Abrufzahlen und damit verbunden die zu produzierenden Stückzahlen, die gegenüber der Serienphase noch gering ausfallen. [31] Nach Nagel et al. [12] gibt es im Wesentlichen drei Zielgrößen, die entscheidend für eine nachhaltige Sicherung der Wettbewerbsfähigkeit im Serienanlauf sind und ein Abhängigkeitsverhältnis zueinander aufweisen. Dabei geht es um den

Kosten-, Zeit- und Qualitätsfaktor. Diese Beziehungen zueinander sind in Abbildung 3.6 dargestellt.

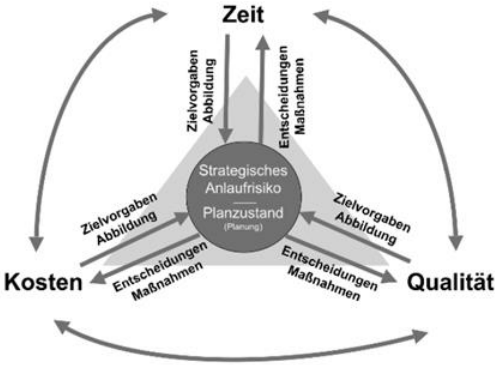


**Abbildung 3.6:** Abhängigkeiten im strategischen Anlaufmanagement nach Nagel [12]

Die drei genannten Eigenschaften weisen eine entgegengesetzt wirkende Zielausrichtung auf, wenn beispielsweise eine Qualitätsmaßnahme viel Zeit in Anspruch nimmt und dadurch Kosten verursacht. Dieser Effekt kann sich jedoch langfristig in der Serienphase komplementär auswirken, wenn z. B. durch ein höheres Qualitätsniveau eine signifikante Fehlervermeidung erreicht wird und somit erneute Qualitätsmaßnahmen weniger zeitintensive Nacharbeit in Anspruch nehmen. Das strategische Anlaufrisiko bildet im Kern bei Entscheidungsträgern eine Risikopräferenz und hilft im Projektablauf einer Serienproduktion bei geplanten Beschaffungs- und Produktionssystemen für ein neues Produkt. [12, 10]

### 3.3.2 Serienphase

Die Serienphase zeichnet sich durch eine stabile und hoch frequentierte Serienproduktion aus. Mit dem Abhängigkeitsverhältnis werden nachfolgend die zentralen Aufwände und Herausforderungen hervorgehoben.

Unternehmen bemühen sich, die von den Kunden gewünschte Produktvielfalt effektiv und wirtschaftlich zu befriedigen, indem sie Plattformorientiert handeln, um Gemeinsamkeiten zwischen Produktreihen, Zielmarktsegmenten und Produktionsprozessen zu identifizieren. Dies stellt die Grundlage für eine effizientere Ressourcennutzung bereit, die bei einer Angebotsvielfalt zu erreichen ist.

Die Varianz der zu produzierenden Erzeugnisfamilien hat auch Einfluss auf die Serienfertigung, wenn beispielsweise eine neue Teilenummer gefertigt werden soll, die eine andere Bestückung aufweist. Hier müssen fertigungstechnische Umrüstvorgänge betrieben werden, die in dieser Zeit die Produktion stillstehen lassen. Daher ist der Nutzungsgrad ein entscheidender Faktor der Serienproduktion.

#### 3.3.2.1 Verschrottung

Zu produzierende Stückzahlen werden durch einen Forecast bzw. eine Bestellmenge vorgegeben. Um schwankenden Rahmenbedingungen, wie Kundenbedarfen, Rüstaufwänden oder Qualitätsproblemen bei Bauteilzulieferern, entgegenzuwirken, werden Ersatzteile oder gar Produkte endbevorratet und eingelagert. Es entstehen Opportunitätskosten in Form der Einlagerung. Langfristig müssen im Zeitraum der Nachserienphase die eingelagerten Ersatzteile oder Produkte verschrottet werden, wenn diese

nicht mehr an den Kunden absetzbar sind. [10, 54]

#### 3.3.2.2 Verfügbarkeit von Bauteilen

Der zeitliche Rahmen zwischen einem Entwicklungsprozess der Phase I und der Einstellung der Produktion EOP in Phase II des PLC beträgt im automobilen Bereich in der Regel fünf bis acht Jahre. Die Kernproblematik ergibt sich daraus, dass Bauelemente-Hersteller zu dem Zeitpunkt des EOP ihre Produkte bereits abgekündigt haben. Falls eine Endbevorratung nicht ausreichend ist oder es keine kompatible Nachfolgegeneration gibt, muss im schlimmsten Fall ein Redesign des Produktes erfolgen. [54]

Um dieser Problematik vorzubeugen, gibt es eine Vielzahl an Nachserienversorgungsstrategien, die im folgenden Abschnitt vorgestellt werden.

## 3.4 Nachserienphase und Auslauf (Phase III und IV)

Das Marktzyklusmodell des Produktlebenszyklus nach Abbildung 3.1 beschreibt die vier Phasen inklusive der Nachserien- und Auslaufphase. Dieses Kapitel beschäftigt sich mit den Nachserienversorgungsstrategien und beschreibt neben der Kernproblematik der Nachserienversorgungspflicht eines Zulieferers auch die zugehörigen Methoden für Ersatzteilbedarfe. Kiritsis et al. verwenden in [13] hierzu ein PLC-Managementmodell, das in Abbildung 3.7 dargestellt ist. Dieses weist Ähnlichkeiten zum vorgestellten Marktzyklusmodell in Abschnitt 3.1 auf und fokussiert hier einen

geschlossenen Regelkreis.

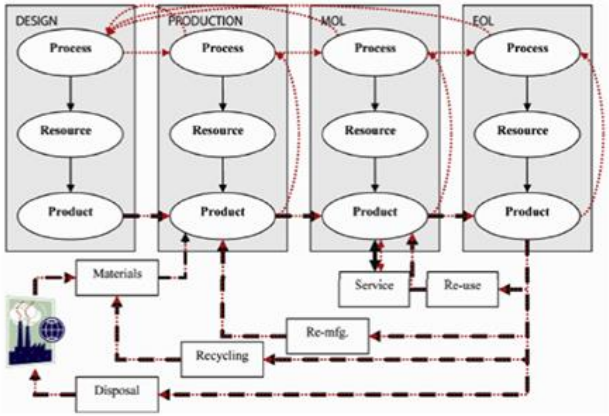


**Abbildung 3.7:** Managementmodell nach Kiritsis [13]

Im Wesentlichen gibt es ebenfalls vier Lebensphasen, die sich aus dem Design, der Produktion, dem Betrieb und der Abkündigung eines Produktes zusammensetzen. Jede Phase besteht aus einem Prozess, einer Ressource und dem Produkt. Entscheidend für die Nachserienversorgungsstrategien sind nun die Möglichkeiten zur Rückführung des Produktes in eine jeweils andere Phase.

So kann ein ausrangiertes Produkt beispielsweise über die Individualreparatur oder eine sonstige Servicedienstleistung wiederverwendet werden. Durch das Remanufacturing wird das Produkt in Analogie zur Serienfertigung wiederaufbereitet und im Markt neu vertrieben. Auch können

Rohstoffe recycelt werden, um über eine Materialaufbereitung neu in eine Fertigung gehen zu können.

Im Folgenden werden die Methoden der Nachserienversorgungsstrategien vorgestellt. Diese bestehen aus der Nachfertigung von Produkten einer bestehenden Serienproduktion, der Endbevorratung sowie Requalifizierungsmaßnahmen eingelagerter Bauteile/Produkte. Ferner werden Nachserienversorgungskonzepte wie das Remanufacturing und die Individualreparatur erklärt. [13]

### 3.4.1 Endbevorratung

Die gängige Praxis zur Deckung künftiger Ersatzteilbedarfe besteht darin, vor Serienabschluss und dem EOP anhand eines Allzeitbedarfes die zu produzierenden Stückzahlen anzupassen. Dies geschieht über Forecast-Konzepte, die eine Prognose der Ersatzteilbedarfe über den gesamten Bereich der Nachserie beschreiben. Entsprechend wird die Fertigung der Bedarfe bereits in der Serienfertigung angepasst. Hier werden auch neben den fertigen Produkten einzelne Komponenten (Bauteilgruppen) endbevorratet. Die Ermittlung und Erstellung von Prognosen über den gesamten Bereich der Nachserie sind ein kritischer Prozess. Die Risiken von Überbzw. Unterdeckungen aufgrund von Fehlprognosen sind hoch und wirken sich finanziell stark aus. [10, 54, 87]

### 3.4.2 Nachfertigung

Ist eine Prognose zu gering kalkuliert, kommt es zu einer Lieferunfähigkeit und einer Vertragsverletzung gegenüber dem Kunden. Hier muss eine Nachfertigung erfolgen. Man differenziert zwischen einer internen und externen Nachfertigung. Die interne Fertigung wird im gleichen Produktionsbetrieb der Hauptserie eingerichtet. Oftmals bildet sich der Bedarf schon vor EOP ab, so dass bereits in der Serienphase eine Nachfertigung etabliert wird.

Die externe Nachfertigung wird bei einem Drittanbieter beauftragt. Der wichtigste Aspekt ist jedoch, dass die Vorgaben des Serienproduzenten die Ersatzteilanforderungen wie Qualitätsansprüche erfüllen. Notwendige Standards der Fertigung und Prüfmitteleinrichtungen sowie die für das Produkt freigegebenen Bauteile müssen eingehalten werden.

Die größten Risiken bilden sich jedoch über die Zeit. So müssen Versorgungszeiträume von in der Produktion verwendeten Bauteilen vorgehalten werden. Ebenfalls muss das Prozess-Knowhow für die Fertigung und deren Anlagen aufrechterhalten werden, was eine zusätzliche Herausforderung für die Mitarbeiterqualifikation darstellt.

Das zentrale Problem für die Nachfertigung ist jedoch die Herstellung von geringen Stückzahlen. So können Betriebsmittel nicht vollständig ausgelastet werden, da großflächig automatisierte Fertigungsbereiche aufgrund von steigenden Betriebskosten nicht zweckmäßig unterhalten werden können. Daher muss ein Redesign der Fertigungsumgebung vorgenommen werden. Somit wird in der Regel eine flexible Auslegung einer Fertigung angestrebt, auf die mit veränderten Bedarfsverläufen in der Produktion reagiert werden kann.

Der Kostenpunkt einer Rekonstruierung ist erheblich und steigert die Produktionskosten, somit auch den Preis des Produktes, aufgrund der geringen Fertigungsmenge und fehlenden Skaleneffekte. Schwankende Bedarfsprognosen bei geringen Mengen können jedoch mit einer Nachfertigungseinrichtung abgefedert werden. [10, 54]

### 3.4.3 Requalifizierung

Im Rahmen einer Endbevorratung oder Einlagerung noch nicht abgerufener Ersatzteile kann es aufgrund der signifikanten Zeitspanne (EOP bis EOD) in der Nachserienversorgung zu Langzeitlagerungsschäden an Bauelementen kommen. Entscheidend hierfür ist die Lagerfähigkeit der Halbleiterkomponenten oder Produkte hinsichtlich der komponentenspezifischen Lagerbedingungen wie Trockenheit und Temperatur.

Auf Grundlage einer ermittelten Zeitspanne muss dann das Produkt/Bauelement requalifiziert werden. Beispielsweise könnte ein Bauteil neu bestromt und einer Kapazitätsmessung unterzogen werden. Bei Speichereinheiten könnten Checksummen des Speicherabbildes kalkuliert und mit den Soll-Werten abgeglichen werden. [54]

Die Ermittlung der Lebensdauerverteilung wird nach Reif [88] über eine Ausfallwahrscheinlichkeit $F(t)$ beschrieben, in der die Betrachtungseinheit zum Zeitpunkt $t_A$ einen Ausfall zeigt. Voraussetzung ist, dass die Betrachtungseinheit funktionsfähig ist mit $F(t = 0) = 0$. Ein Ausfall über

den Zeitpunkt tA ergibt einen Ausfall mit $F_(t) = 1$. Die Zuverlässigkeitsfunktion $R(t)$ ist nach [88] in Gleichung 3.1 beschrieben.

$$\begin{aligned} Es\ gilt\ F_{(t)} \ +\ R_{(t)} \ =\ 1 \\ mit\ F_{(t)} \ =\ 0\ f\ddot{u}r\ t < t_A\ zeigt\ noch\ keinen\ Ausfall \\ und\ F_{(t)} \ =1\ f\ddot{u}r\ t > t_A\ zeigt\ einen\ Ausfall \end{aligned} \tag{3.1}$$

Die Ausfallwahrscheinlichkeit sowie die Lebensdauerverteilung $F(t)$ können experientell nicht exakt bestimmt werden. Daher approximiert man die Zuverlässigkeitsfunktion $R(t)$, indem man die Änderung einer Wahrscheinlichkeit, ein System ausgefallen anzutreffen, mit der Wahrscheinlichkeitsdichte der Lebensdauer $f(t)$ aus Gleichung 3.2 in Beziehung setzt.

$$f_{(t)} = \dot{F_{(t)}} \tag{3.2}$$

Prinzipiell finden sich Requalifizierungsmaßnahmen auch in der Wiederaufbereitung eines Produktes.

### 3.4.4 Remanufacturing

OEMs streben eine frühe Einstellung der Ersatzteilproduktion an. Produktionskosten steigen aufgrund sinkender Produktionsleistung bei geringerem Volumen. Auch werden die Ressourcen für neuere Produktionslinien benötigt. Die Aufbereitung von Kraftfahrzeugen und deren Steuergeräten gewinnt daher im automobilen Bereich vermehrt an Bedeutung. Zudem spielt die gesellschaftliche Entwicklung eine maßgebende Rolle. Neben der Gesamtzahl der zugelassenen Autos in Deutschland steigt ebenfalls

deren Lebensalter. Mit längerer Nutzung erhöht sich allerdings auch die Ausfallwahrscheinlichkeit F(t). [14]

Ein Remanufacturing-Prozess ist in der Regel ein Wiederaufbereitungsprozess, in dem das Produkt als Schadteil aus dem Feld kommt und es qualitativ nach serienproduktionstechnischen Standards wiederaufbereitet wird. Die entscheidenden Charakteristika bestehen darin, dass das Schadteil aus einem fehlerhaften System herausgenommen wurde und es so rückgesetzt wird, dass es einem neuen System zugeführt werden kann. Der Aufbereitungsprozess lässt sich nach Casper [14] in eine offene oder geschlossene Lieferkette unterteilen. Beide Modelle sind in Abbildung 3.8 dargestellt.

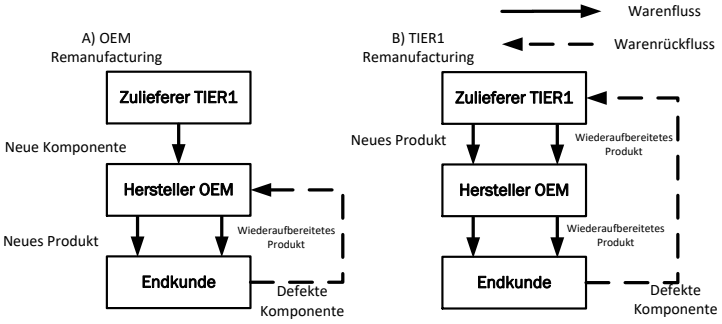


**Abbildung 3.8:** Modell der geschlossenen Lieferketten nach Casper [14]

Abbildung 3.8 A) beschreibt, dass beispielsweise ein OEM ein neues Produkt von einem Zulieferer (TIER1) erhält und an den Kunden vertreibt. Bei Ausfall entnimmt der OEM das Schadteil, bereitet es selbst auf und

vertreibt es im gleichen Pool an neue Kunden. Dieser Warenfluss wird als geschlossene Lieferkette bezeichnet.

Abbildung 3.8 B) beschreibt die geschlossene Lieferkette, in der der Zulieferer und auch Hersteller des Produktes selbst den Wiederaufbereitungsprozess übernimmt. Dies bedeutet, dass oftmals Schadteilsammlungen vom OEM erstellt und dem Zulieferer zur Verfügung gestellt werden. Der OEM kann nun aus einem Pool auswählen, ob er ein neues oder wiederaufbereitetes Gerät haben möchte, und übergibt es den Kunden. Nach Xiong [89] ist diese Versorgungsstrategie die gängige Praxis, da heutzutage nur wenige OEMs das Produkt selbst entwerfen und produzieren.

Spohie et al. beschreiben in [90] die Vorzüge von wiederaufbereiteten Produkten. Ein Wiederaufbereitungsprozess ist allgemein abhängig von der Wiederaufbereitungsquote und der Wiederaufbereitungsmenge. Analog zu der Serienfertigung besteht hier ebenfalls die Herausforderung der Vorhaltung von Bauelementen für das Produkt. In der Regel sind die Prozesskosten weitaus geringer als die Herstellungskosten eines neu anzufertigen Produktes. Außerdem lassen sich schwankende Abrufzahlen besser puffern, da ein Remanufacturing-Prozess gegenüber einer Serienfertigung eine höhere Flexibilität bietet.

### 3.4.5 Individualreparatur

Neben dem Remanufacturing ist die Individualreparatur die populärste Art einer Wiederaufbereitung. Die Individualreparatur ist im Independent Aftermarket (IAM) platziert und hat eine direkte Zulieferkette vom Dienstleister zum Kunden. Das entscheidende Charakteristikum ist, dass das Produkt nach erfolgter Reparatur wieder im selben System eingebaut wird.

Sämtliche individuellen Daten bleiben erhalten. Auch hat der Qualitätsanspruch andere Bedürfnisse. Während ein Reman-Produkt die gleichen Anforderungen wie ein Neuteil einhält, wird in der Individualreparatur zielgenau der Fehler behoben, um einen erneuten Betrieb zu ermöglichen. Dies erlaubt auch die Wiederaufbereitung eines Produktes weit nach dessen Abkündigung (EOD). [91]

Der Reparaturprozess selbst ist bestimmt von der Schwere des Schadens. Üblicherweise existieren abhängig vom Dienstleister individualisierte Reparaturverfahren. Die Herausforderung ergibt sich einerseits durch die Produktvielfalt, andererseits auch infolge der Komplexität selbst. So ist beispielsweise ein Motorsteuergerät thermisch versiegelt, was ein zerstörungsfreies Öffnen des Gehäuses oftmals erschwert. Des Weiteren offenbaren die Halbleiterbauteile eine große Vielfalt an Bauteilvarianten. Reparaturverfahren erfordern daher oftmals professionelle Ausrüstung, wie aufwendige BGA-Lötstationen oder Sondervorrichtungen. Auch sind Steuergeräte softwaretechnisch proprietär und stark geschützt. Sie bieten in der Regel nur standardisierte Diagnosefunktionen über die OBD-II-Schnittstelle. [91]

Für die Förderung einer ressourceneffizienten Kreislaufwirtschaft gibt es nachhaltige Reparaturstrategien, die im Kontext des Industrie-4.0-Ansatzes neue Wiederherstellungsstrategien der Unternehmen fordern. [92]

## 3.5 Variantenmanagement in einer Serienproduktion

Unternehmen versuchen wirtschaftlich, die am Markt geforderte Vielfalt von Produkten zu bedienen. Plattformkonzepte helfen bei der Identifikation

von Produktionsprozessen und setzen die Ressourcennutzung bei der Angebotsvielfalt um. Kernproblem ist jedoch, dass künftige Produktdesigns häufig die Modifizierung und Überarbeitung eines Konzeptes erfordern. [83, 87, 93, 94]

Fujita stellt in [95] drei Hauptproblematiken dar, die sich aus der Optimierung der Produktmodule mit festen, vordefinierten und kombinierten Modulattributen ergeben. Wenn die Gemeinsamkeit von Modulen und Varianten einer Plattform zunimmt, steigt auch das Produktionsvolumen. Mit steigender Anzahl der Varianten oder bei Änderungen gemeinsamer Module wird das Produktionsvolumen verringert und verursacht Kosten aufgrund zu hoher Spezifikationen. Produktplattformen sollten daher aus technologischer und architektonischer Sicht über eine möglichst lange Zeit erhalten bleiben. Eine zu lange Stabilitätsperiode verhindert jedoch Innovationen. Dynamische Produktvarianten spielen daher für das Plattformdesign eine entscheidende Rolle, um die Anpassungsfähigkeit von Plattformmodulen, die Änderungen unterliegen, zu erhöhen. [83, 96, 97]

Nach Zhang werden in [98] Module in vier Bereiche gegliedert und unterteilen sich in (1) Module mit festem Kern und gemeinsamen grundlegenden Kundenanforderungen, (2) neue Module für neue Anforderungen, (3) kundenspezifische Module und (4) Module in Bezug auf Parameter der physikalischen Struktur.

Das Definieren von separaten Produktvarianten führt zu redundanten Daten. Hallerbach et al. schlagen in [99] eine Möglichkeit vor, durch eine hierarchische Struktur aus einem Basisproduktmodell neue Module einzugliedern, deren Eigenschaften bereits vorhanden sind und übernommen werden, ohne redundante oder inkonsistente Daten zu erstellen. Somit kann eine große Sammlung verwandter Varianten abgeleitet werden, wobei jede Variante eine Anpassung des Grundmodells ist.

Ausgehend vom Variantenmodell hat Bormann [10] am Beispiel eines Motorsteuergerätes ein Variantenmanagement beschrieben, das erheblich zu der Vermeidung der Schrottkosten und anderen Einsparpotentialen beiträgt. Hinzu kommen Produktionsprozesse, bei denen eine wiederholte Inbetriebnahme mit aktualisierter Softwareversion vermieden wird.

Die Kernproblematik ergab sich aus der Verwendung von OTP-Restriktionen aus Abschnitt 2.2.1. Aufgrund der Fertigstellung eines Serienproduktes war eine Umflashmaßnahme zu einer anderen Variante mit hardwarekompatibler Varianz nicht mehr durchführbar. Es mussten somit regelmäßig Verschrottungen vorgenommen werden, da Inbetriebnahmen aufgrund einer nicht aktuellen SW-Version nicht möglich waren.

Im Wesentlichen wird im Lösungsaspekt aus einer Erzeugnisfamilie ein Hardware- und Softwarevariantenmodell verwendet. Im Modell werden Module mit physikalischen Strukturen und verschiedenen Softwareversionierungen differenziert. Abbildung 3.9 zeigt das Variantenmodell.

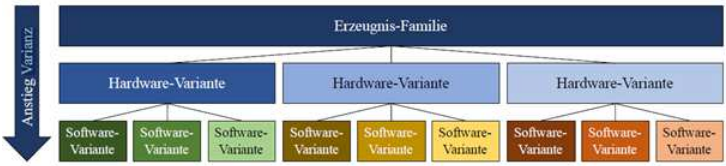


**Abbildung 3.9:** Variantenmodell nach Bormann [10]

So sollen bei Produktionneuanläufen Rohlingstypen eingesetzt werden, die den Hardwarevarianten entsprechen. Sollten simultan Softwareänderungen

erfolgen, können die zwischengelagerten Produkte einfach auf die finale (Software-)Variante umgeflasht werden.

## 3.6 Fazit

Die beschriebenen Prozesse zur Nachserienversorgung leisten einen wichtigen Beitrag für den Produktlebenszyklus. Aus Sicht der erläuterten Security- und Safety-Anforderungen aus Abschnitt 3.2.1.2 und 3.2.1.3 bleibt jedoch die Herausforderung erhalten, die Rücksetzung eines Systems respektive eines Produktes und dessen Softwareaktualisierung zu gewährleisten. Dies hat auch Auswirkungen auf das beschriebene Variantenmanagement in Abschnitt 3.5. Bereits finalisierte Geräte bleiben im Rahmen eines Aufbereitungskonzeptes aus Abschnitt 3.4.4 nicht mehr umflashbar, wenn sich die Signierungsquelle einer Software ändert. Das folgende Kapitel beschäftigt sich mit dieser Kernfrage des Lösungsaspektes.

# 4 Problemanalyse

In den beiden vorausgegangenen Kapiteln wurden Grundlagen über kryptografische Absicherungsmaßnahmen zum Schutze der Integrität und Authentizität einer Software und von deren angewandten Einsätzen im Produktlebenszyklus des automobilen Umfelds behandelt.

Weiterführend soll nun das aufkommende Kernproblem der Updatefähigkeit einer Software mit neuer Autorität beschrieben werden, um aufgrund des Handlungsbedarfes die Forschungsfragen ableiten zu können.

Dieses Kapitel betrachtet daher zunächst Absicherungsmaßnahmen zum Schutze der Integrität und Authentizität einer Software aus anderen Domänen wie dem Mobilfunkbereich oder sonstigen Anwendungen im Bereich Internet of Things (IoT).

Anschließend wird ein praxisnahes Beispiel aus dem Car-Multimediabereich das Grundproblem beschreiben. Hierbei wird ein Durchlauf einer Head Unit durch den gesamten Lebenszyklus betrachtet, da die Umsetzung von Produktanforderungen aus dem Security- und Safety-Bereich in das Produkt einen bedeutenden Einfluss auf die Prozessstruktur in der Serienfertigung ausübt. Darüber hinaus werden die Auswirkungen der Nachserienversorgung in Bezug auf die Updatefähigkeit der Software explizit dargestellt, um das Problem kontextspezifisch zu erläutern.

Daraus resultiert im dritten Abschnitt der Handlungsbedarf, der die Kernproblematiken anhand des praxisnahen Beispiels beschreibt. Daraus wird im vierten Abschnitt die Forschungsfrage abgeleitet und somit die Zielsetzung dieser Arbeit festgelegt.

## 4.1 Bekannte Anwendungen aus anderen Plattformarchitekturen

Wenn ein Produkt erstmals in einen Initialzustand gestellt wird, beinhaltet dies neben der Installation einer Software auch die Bildung und Aktivierung einer Vertrauensbasis. Diese wird einmalig in Form eines asymmetrisch-kryptografischen Schlüsselpaares generiert, das die Autorität auch für weitere Softwareversionen des Produktes oder der Produktreihe darstellt. Aus technischer Sicht beinhaltet die Installation einer Software neben der Speicherung des Codeblockes in das Produkt eine Authentifizierung und Integritätsprüfung. Über den zu installierenden Codeblock wird eine Hashberechnung durchgeführt. Der private Schlüssel verschlüsselt nun den Hashwert des Codeblockes und bildet somit eine Signatur. Der für die Authentifizierung benötigte öffentliche Schlüssel wird in der Regel in einem einmalig beschreibbaren Speicher im Produkt platziert. Zum Starten und Betreiben der Software wird im Produkt nochmals ein Hashwert des Codeblockes berechnet. Parallel wird die Signatur mit dem öffentlichen Schlüssel entschlüsselt. Es ergibt sich somit wieder der Hashwert des Codeblockes. Beide Hashwerte, die des Codeblockes und der entschlüsselten Signatur, werden miteinander verglichen. Hier darf es keine Abweichungen geben, da sonst der Codeblock verändert sein könnte oder die Signierung des Codeblockes einer anderen Autorität entsprechen könnte. Durch das irreversible Ablegen des öffentlichen kryptografischen

Schlüssels in das Produkt kann nur noch eine Software von derselben Autorität betrieben werden. Die Authentifizierung und Integritätsprüfung werden durch den Bootloader vorgenommen. Häufig wird Software vollständig vom OEM oder von dritten Parteien entwickelt. Die Signierung der Software wird jedoch meist vom Produkthersteller (Zulieferer) mit Genehmigung des OEM durchgeführt. Langfristige Konzepte für die Verteilung der Sicherheitsupdates des Produktes müssen hier bereitgestellt werden.

Der folgende Abschnitt betrachtet nun andere Produktbereiche. Hier werden die Systemsicherheitskonzepte kurz diskutiert und mit dem Ansatz der vorliegenden Grundlagen in Beziehung gesetzt.

In der Unterhaltungselektronik verwendete Erzeugnisse arbeiten größtenteils mit Derivaten der Arm-Architektur. Ein vorgegebener Startprozess sorgt für eine sogenannte „chain of trust“, einen sicheren Bootvorgang, der jede Komponente (Codeblock) vor der Ausführung authentifiziert. Die Authentifizierung geschieht über die bekannte Methodik einer Überprüfung der Signatur mittels eines öffentlichen Schlüssels. Das Eigentum (Intellectual Property) könnte sich jedoch mit jeder Bootphase ändern. So könnte ein Bootloader zu einem OEM gehören und die Zielapplikation, also ein eigenständiger Codeblock, zu einer anderen Partei. Sinngemäß müsste sich hier die Authentifizierungsquelle durch einen sekundären öffentlichen Schlüssel ändern. Dies greift jedoch in die Grundlagen eines definierten Secure Boot ein. Der öffentliche Schlüssel kann nur als vertrauenswürdig angesehen werden, wenn ausgeschlossen ist, dass dieser trivial modifiziert werden kann. Reprogrammierungs- und Seitenkanalangriffe oder ein Reengineering müssen ausgeschlossen werden. Üblicherweise ist das „System on Chip“-ROM die einzige Komponente im System, die eine Möglichkeit des One Time Programmable (OTP) bereitstellt. Dieses speichert eindeutige Werte irreversibel. [100] Für die Sicherheit eines

Systems auf Applikationsebene werden zusätzliche Partitionierungen der Hardware- und Softwareressourcen des SoC verwendet. Hier spricht man in der Arm-Architektur von einer Trust Zone. Die Separation einer sicheren Umgebung wird durch eine Hardwarelogik gewährleistet. Diese ist im Arm-Prozessorkern (Core) verankert. Ergänzungen im Core ermöglichen es, Codes aus der sogenannten normalen Umgebung zur sicheren aufzutrennen. Hierdurch wird ein dedizierter Prozessorkern erspart. Jeder physische Prozessorkern bietet zwei virtuelle Kerne. Der Sicherheitsstatus ist auf dem internen Systembus codiert und ermöglicht eine Integration der virtuellen Prozessoren in den Systemsicherheitsmechanismus. Der als nicht vertrauenswürdig eingestufte virtuelle Prozessor kann nur auf freigegebene Systemressourcen zugreifen, während der vertrauenswürdige alle Ressourcen sehen kann. [100, 101]

Lösungskonzepte anderer SoC-Architekturen sind auf die jeweiligen Einsatzfelder ausgelegt. TriCore-Architekturen der Firma NXP, die in sicherheitskritischen Bereichen eingesetzt werden, verwenden ein sogenanntes Hardware-Security-Module – kurz HSM. Der Ansatz bestand darin, sicherheitsempfindliche Vorgänge in dedizierte und in sich geschlossene Sicherheitssubsysteme auszulagern. Ein HSM ist ein hardwarebasiertes und in sich abgeschlossenes Subsystem und weitgehend für die Verwaltung/Verarbeitung kryptografischer Schlüssel ausgelegt. Viele HSM sind in der Verwendung für Manipulationserkennungen eingesetzt und reagieren funktional zu der im SoC befindlichen Peripherie. Auch hier ist ähnlich wie bei der Trusted Platform eine physische Trennung von sicherheitsrelevanten Operationen und dem Rest des Host-Systems gegeben. [101]

### 4.1.1 Endgeräte am Beispiel des iPhones

Die Systemsicherheit bei Mobilgeräten hat durch das große Marktpotential einen besonderen Stellenwert und öffentliches Interesse. Hier liegt der Fokus nicht nur darauf, die im Gerät gespeicherten Daten zu schützen, sondern auch auf der gesamten Umwelt, also allem, was der Anwender in Applikationen über Netzwerke und mit Internetdiensten ausführt. Zentral gesetzte Sicherheitsfunktionen (z. B. die Geräteverschlüsselung) sind so ausgelegt, dass sie sich nicht vom Anwender versehentlich deaktivieren lassen, ohne die Benutzerfreundlichkeit einzuschränken.

Um die Integrität zu gewährleisten, werden alle Systemkomponenten auf Vertrauenswürdigkeit validiert. Diese Methodik basiert auch hier auf der Signierung der einzelnen Komponenten und der Absicherung eines Apple-Root-Zertifikates im Boot-ROM-Code. Erst nach der Verifizierung des jeweiligen Schrittes wird zum nächsten Codeblock übergegangen. Zu den signierten Komponenten gehört die Firmware, wie der Bootloader, der Kernel, Kernelerweiterungen und die Baseband-Firmware. Die Software- und Sicherheitsaktualisierungen müssen gerade im Multimediabereich, aber auch bei mobilen Endgeräten gegeben sein. Hier muss zudem sichergestellt werden, dass nicht auf alte Betriebssystemversionen zurückgegriffen wird. Angreifer könnten diese nutzen, um eine ältere Version mit bekannten Sicherheitslücken zu kompromittieren.

Auch aufgrund der Gegebenheit, dass sämtliche Softwareupdates fast ausschließlich „over the Air“ erfolgen, gelten hier besondere Anforderungen. Im Updateverfahren wird eine Verbindung mit dem Apple-Server für die Installationsautorisierung hergestellt. Kryptografische Kennzahlen (Zufallswerte und eindeutige Kennung des Gerätes [ECID]) werden für jeden

Teil eines Installationspakets (Kernel, OS-Image etc.) gesendet. Die Versionen werden vom Apple-Autorisierungsserver intern mit einer gültigen Liste verglichen die aufzeigt, welche Installation für das Endgerät erlaubt ist. Bei einer Übereinstimmung der ECID mit den Kennzahlen wird die Installation freigegeben. [102]

### 4.1.2 Plattformen

Die Sicherheit von IoT-Plattformen wird vermehrt diskutiert. Oftmals werden diese in Zusammenhang mit sogenannten Bot-Netzwerken gestellt. Unabhängig von der verwendeten Technologie der Rechnerarchitektur liegen die größten Risiken in der Applikation (App), Infrastruktur und den Kommunikationswegen. Wie mobile Endgeräte sind auch die IoT-Geräte physisch erreichbar. Dies bedeutet, dass die Sicherheit zur Laufzeit beeinträchtigt werden kann. [103]

Nach S. J. Johnston [9] wird eine Differenzierung von vier Security-Graden erstellt, sie ist in Abbildung 4.1 veranschaulicht. Diese beinhalten die Verschlüsselung des im Gerät befindlichen Speichermediums, wie einer SD-Karte, bis hin zum signierten und verschlüsselten Bootloader. Im Grunde wird auch hier der Ansatz verfolgt, dass ein Boot-ROM die Bootloader-Signatur überprüft, um somit eine vertrauenswürdige Bewertung der Komponenten zu gewährleisten. [15]

Das erste Level verfolgt den Ansatz einer Vollverschlüsselung des Speichermediums. Hierzu werden gängige Verfahren verwendet, wie sie auch im Desktopbereich eingesetzt werden. Dies erfordert üblicherweise eine Passworteingabe beim Startvorgang. Es gestaltet sich jedoch zum Nachteil,

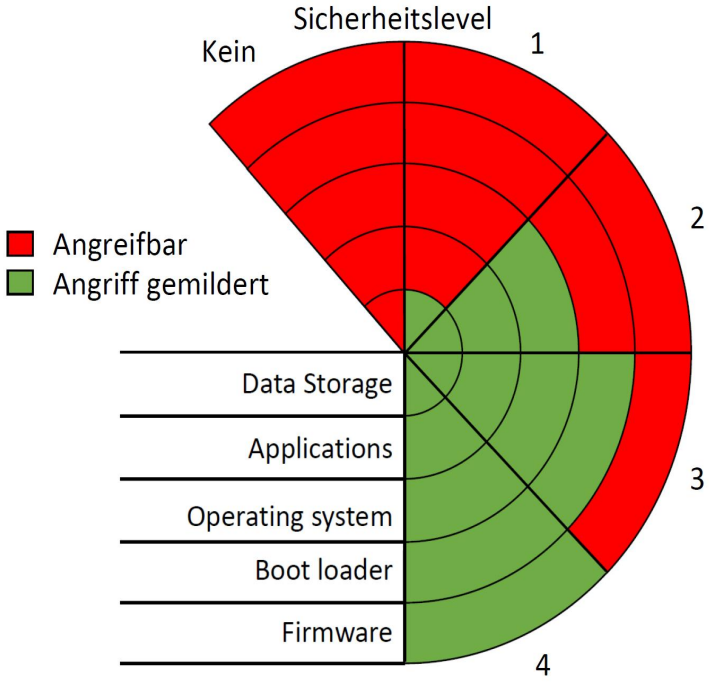


**Abbildung 4.1:** Sicherheitslevel von IoT Platformen nach [15]

da IoT-Geräte oftmals an abgelegenen Orten eingesetzt werden. Durch das Speichern eines Passwortes auf einer unverschlüsselten Partition wird dieser Schutz somit aufgehoben, wenn ein Angreifer physischen Zugriff auf ein Gerät hat. Das zweite Level beschreibt die Überprüfung der Systemintegrität einer Applikation und des Operationssystems durch einen Bootloader. Dies stellt auch höhere Anforderungen an die eingesetzte Rechnerarchitektur, da hier kryptografische Funktionen benötigt werden. Der notwendige Key für die Verschlüsselung wird vom Bootloader verwaltet. Dafür stellen gängige Rechnerarchitekturen ein sogenanntes „Trusted Platform Module

(TPM)“ bereit. Der Key wird hier verschlüsselt abgelegt und kann nur über die Funktionalität der TPM durch den Bootloader zum Entschlüsseln der Partitionen verwendet werden. Das dritte Level sieht als weiteren Schritt vor, dass der Bootloader durch ein asymmetrisches Schlüsselpaar signiert und der öffentliche Schlüssel irreversibel auf der Rechnerplattform gespeichert wird. Das Boot-ROM der Rechnerplattform überprüft die Signatur des Bootloaders beim Startvorgang. Somit ist im Gegensatz zum zweiten Level ein Missbrauch des Bootloaders ausgeschlossen, beispielsweise wenn dieser kompromittiert oder übergangen wird. Das vierte Level sieht zudem vor, den Bootloader-Code zu verschlüsseln. Die Funktionalität zum Passwort-Handling muss hierbei der Boot-Rom-Code bereitstellen.

Ein 2017 verfasstes Statement des Verbandes der Automobilindustrie (VDA) zu Security-Maßnahmen und -Prozessen nach [104] unterstützt berechtigte gesellschaftliche Interessen von ausreichend hohem Security-Niveau. Hochgradig standardisierte Produkte mit abgegrenzten Systemen, klaren Sicherheitsanforderungen und regulierten Märkten zeigen eine starke Best Practice bei Zertifizierungen nach der Common Criteria (CC). Die Produkte in der Automobilindustrie zeichnen sich durch stetig weiterentwickelte Funktionsumfänge sowie OEM-spezifische Sicherheitsarchitekturen und Prinzipien aus. Der Ansatz einer zentralen Zertifizierungsstelle ist dem VDA zufolge für solch heterogene und komplexe Systeme innovationshemmend und kostenintensiv. Optimierte Lösungen können ggf. nicht konform wirken und müssen im Einzelfall betrachtet werden. Verpflichtende Zertifizierungen bergen die Gefahr, dass Sicherheitsprofile durch eine lösungskonforme Auslegung beschränkt werden. Somit ergeben sich eine trügerische Sicherheit und das Risiko, dass angepasste Maßnahmen nicht der tatsächlichen Security der Systeme entsprechen. Auch zum Schutz der Besitzer, Fahrer und Insassen der Fahrzeuge wird eine verpflichtende Zertifizierung nicht empfohlen. Regulierungen und die Festlegung von

technischen Maßnahmen kommen nur dann in Frage, wenn die Ziele der Anforderungen nicht vom Markt selbst erreicht werden. Konkrete Lösungskonzepte zur Erfüllung der Ziele sollen durch den Hersteller individuell im Kontext der Gesamtsystem-Security-Architektur umgesetzt werden.

## 4.2 Kryptografische Absicherungsmaßnahmen im Lebenszyklus anhand eines Infotainment-Systems

Das folgende Praxisbeispiel bezieht sich auf das Infotainment-System einer sogenannten Head Unit, auch A-IVI genannt. A-IVI steht für Alliance In-Vehicle Infotainment. Der Produktlebenszyklus des Gerätes wird in Analogie des aus Abschnitt 3.1 im Folgenden durch den fertigungstechnischen Entstehungsprozess, der Updatemaßnahmen und den Wiederaufbereitungsprozess beschrieben. Dabei wird zunächst auf die gegebene Rechnerarchitektur im Sinne des Hardware-Security-Moduls und dessen Handling-signierter Software im Bootvorgang eingegangen. Anschließend folgt der Fertigungsprozess, wo je nach Durchlauf der verschiedenen Teststufen die kryptografischen Eigenschaften und Sicherungen aktiviert werden. Weiterhin werden Softwareupdatemaßnahmen nach der Produktion erläutert, die sowohl im Werk als auch im Feld vollzogen werden. Final werden der Wiederaufbereitungsprozess und die darin enthaltenen Herausforderungen erörtert.

### 4.2.1 Rechnerarchitektur

Die im Infotainment-System integrierte Rechnerarchitektur basiert im Kern auf dem i.MX6 der Firma NXP. Diese Mikrocontrollerfamilie lehnt sich an die Cortex-A9-Arm-Architektur an, wurde proprietär erweitert und verfügt über systemspezifische Einrichtungen wie das Power und Memory Management, aber auch Security-Institutionen wie den High Assurance Boot (HAB). [16]

#### 4.2.1.1 High Assurance Boot

Der High Assurance Boot besteht aus einer HAB-Bibliothek, die eine Unterkomponente des Boot-ROMs im i.MX6-Prozessor darstellt, und dem Super-Root-Key-Register (SRK-Register). Der HAB ist verantwortlich für die Überprüfung der digitalen Signaturen der zu laufenden Software. Vom HAB bereitgestellte Funktionen werden im Rahmen der Bootchain vom Bootloader oder von anderen Softwareinstanzen verwendet. Es wird sichergestellt, dass nur ein authentifizierter Code ausgeführt werden kann, wenn der Prozessor als sicheres Gerät (Secure Device und Secure Execution aus 2.3.2) konfiguriert ist. Hierbei wird eine signierte Software benötigt, um die Integrität von Software außerhalb der Prozessorumgebung zu schützen. In der Regel wird mit der Ausführung eines Bootloaders begonnen.

Der signierte Codeblock einer Applikation oder eines Bootloaders beinhaltet neben verschiedenen Mapping- und Instruktionstabellen ein „Config Structure File“ (CSF), bestehend aus einer Signatur und einem Zertifikat, wo der zugehörige öffentliche Schlüssel abgelegt ist. Die Authentifizierung des signierten Codeblockes durch den HAB erfolgt mit dem bekannten

Verfahren aus 2.3.1. Vorbedingung ist jedoch die Validierung des im CSF enthaltenen öffentlichen Schlüssels.

Hier muss als Vorbedingung ein sogenannter „Super Root Key“ im sogenannten SRK-Register (ein Fuse-Register nach 2.2.1) irreversibel abgelegt sein. Dieses Register kann einmalig bis zu vier Hashes (SRK1–SRK4) aufnehmen. Die HAB-Einrichtung kann diese über ein sogenanntes Shadow-Register auslesen. Abhängig von der Konfiguration des HAB wird nun ein SRK beim Bootvorgang selektiert. Dies bedeutet den Startpunkt zu einer definierten Adresse im Flash-Speicher, wie er im Abschnitt 2.5.1 unter Abbildung 2.8 angezeigt ist. Für die Bootfolge sind die signierten Codeblöcke nur mit den selektierten SRKs gültig. Ebenfalls ist es möglich, einen SRK über den HAB für kompromittiert zu erklären. Hierfür gibt es das „Revocation-Fuse-Register“, das bis zu drei Super Root Keys für ungültig erklären kann und es auch sicherstellt, dass bei einem gesetzten Secure Device mindestens ein SRK gültig bleiben muss.

Der HAB berechnet einen Hashwert des öffentlichen Schlüssels aus dem Config Structure File und vergleicht ihn mit dem hinterlegten Hashwert des Super Root Keys. Ist der SRK gültig, erfolgt nun die Entschlüsselung der Signatur und der Vergleich der Hashwerte des Codeblockes in Analogie des Secure-Boot-Prozesses aus 2.3.1. Sollte an einer Stelle eine Unstimmigkeit oder ein falsches Ergebnis ermittelt werden, so kann die überprüfende Softwareinstanz darauf reagieren und ggf. den Bootprozess einschränken oder in einen speziellen Software-Resetstatus gehen.

#### 4.2.1.2 Bootvorgang

Der Bootloader oder eine Zielapplikation ist in getrennten Codeblöcken, mit einer definierten Adresse im System-Adressbereich strukturiert. Das signierte Image eines Bootloaders und ggf. einer Applikation wird auf dem i.MX-Prozessor unter Verwendung der entsprechenden öffentlichen Schlüssel verifiziert. Abbildung 4.2 A) zeigt schematisch den Bootprozess einer Zielapplikation. Das Boot-ROM des i.MX6 bewertet mit Verwendung der HAB-Bibliothek die Signatur der Zielapplikation im Hinblick auf die Gültigkeit. Bei positiver Validierung wird nun die Applikation gestartet.

Andernfalls wird ein Error State aktiviert und es erfolgt eine gesonderte Prozedur. Gleiches gilt, wenn neben der Zielapplikation noch ein Bootloader als Zwischeninstanz gebootet wird, siehe Beispiel B) in Abbildung 4.2. Hierbei erfolgt die gleiche Prozedur innerhalb der Bootchain. Der Bootloader bewertet mit Hilfe der HAB-Bibliothek die Signatur der nächsten Instanz, die hier die Zielapplikation bildet. Dabei wird zwingend nur ein SRK (Super Root Key) für sämtliche Images verwendet. Der Bootloader und die Applikation haben die gleiche Autorität.

#### 4.2.1.3 Master Key

Unabhängig vom sicheren Bootvorgang ist es oftmals auf Applikationsebene nötig, Datensätze zu sichern und verschlüsselt abzulegen. Die Architektur des i.MX6 verfügt über einen sogenannten Master Key (MK). Dieser Key ist fest als Geheimnis in Analogie zu einer gerätespezifischen Eigenschaft aus Abschnitt 2.2.2.1 in der CPU hinterlegt und kann nicht ausgelesen werden. Hiermit werden beispielsweise Berechtigungslevel auf Applikationsebene etabliert, indem ein Zertifikatsmanagement verwendet

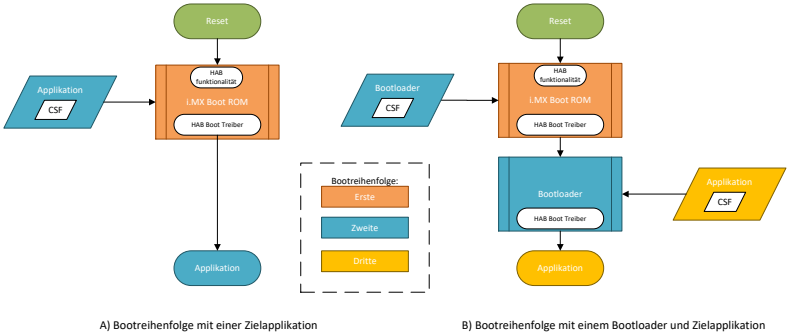


**Abbildung 4.2:** Secure Boot Chain i.MX6 nach [16]

wird. Auch werden hier Passwörter oder Freischaltcodes, die auf Applikationsebene eingesetzt werden, verschlüsselt abgelegt. Die HAB-Einrichtung ermöglicht somit eine Funktionalität, Codebereiche oder eigene Datensätze wie separate Zertifikate verschlüsselt auf einem Datenträger abzulegen. Dieser verschlüsselte Datenbereich wird als Secure Device Container (SDC) bezeichnet und kann einen Speicherbereich im externen Flash verwenden.

### 4.2.2 Der Fertigungsprozess

Im folgenden Abschnitt wird nun der Fertigungsprozess erläutert und auf die jeweiligen Prozessschritte eingegangen. Es werden folgend die Bestückung, Montage der Leiterplatte und der Endprüfung des gesamten Produktes betrachtet. Abbildung 4.3 zeigt den sequenziellen Fertigungsverlauf der Leiterplatte (A) und des komplett montierten Gerätes (B).

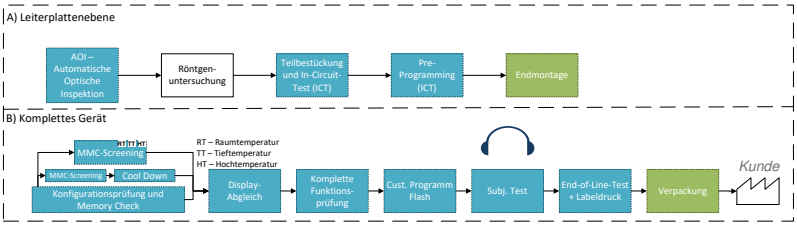


**Abbildung 4.3:** Fertigungsprozess des A-IVI

#### 4.2.2.1 Bestückung der Leiterplatte

Zunächst wird die Leiterplatte einer automatischen und optischen Inspektion unterzogen. Komponenten wie der Hauptprozessor, Treiberbausteine und Speicherelemente werden in der Teilbestückung auf die Leiterplatte gelötet. Aufgrund der enormen Datenmengen von 64 GB wird das externe Flash, der eMMC, bereits „vorgeflasht“ und im bereits geflashten Status eingelötet. Zur Absicherung wird ein sogenannter In-Circuit-Test vollzogen, der Innenwiderstände und Kapazitäten im Netzwerk überprüft und somit den Fertigungsprozess sicherstellt.

Nach diesem Prozess kommt es zur ersten Bedatung der externen Speichereinheiten. Hier wird vom Pre-Programming gesprochen, das ebenfalls In-Circuit vorgenommen wird. Hierbei werden die auf der Leiterplatte befindlichen Controller über eine Nadelkontaktierung angeschlossen und restliche Konfigurationen vorgenommen. Das Softwareimage beinhaltet die signierte Zielapplikation, volatile Datensätze, Kartenmaterial, Audiofiles und den Bootloader. Eine separate Testsoftware-Umgebung ist parallel implementiert. Diese wird im Rahmen des Fertigungsprozesses eingesetzt

und verwendet ebenfalls eine signierte Applikation, jedoch mit einer anderen Autorität. Die Bereitstellung erfolgt in Analogie zu der Sicherheitsintegration aus Abschnitt 3.2.1.5.

Im Pre-Programming werden somit zwei SRKs für die Zielapplikation (SRK1) und die Fertigungs-Diagnosesoftware (SRK2) gesetzt. Zudem wird über den HAB der Prozessor als sicheres Gerät (Secure Device) konfiguriert. Die beiden verwendeten SRKs werden irreversibel gesetzt. Es folgt die Endmontage des Infotainment-Systems. Neben der Anbringung sämtlicher Steckerleisten wird nun die Leiterplatte in ein Gehäuse gefasst und auch das Display mit der Hauptplatine verknüpft. Es folgt nun die komplette elektrische Prüfung und Finalisierung des Gerätes.

#### 4.2.2.2 Fertigung des kompletten Gerätes

Zunächst wird ein sogenanntes Screening durchgeführt, das einen „Burn-In-Test“ darstellt. Somit reduzieren Frühausfälle die Ausfallrate des Halbleiters beim Kunden. Hierbei wird das Gerät bestromt und Funktionen der Fertigungs-Diagnosesoftware werden ausgeführt. Die Tests sehen einen kompletten Memory Check vor und prüfen zudem die Konfigurationseinstellungen des Gesamtsystems. Anschließend wird ein Displayabgleich durchgeführt. Eine darauf folgende elektrische Endprüfung testet alle Komponenten auf Funktion. Hierzu zählen Lüftereinheiten, GPS, Audio, Performance, Stromaufnahme und Kommunikation. Anschließend wird nun der finale Softwarestand konfiguriert. Diese Konfiguration beinhaltet spezifische Einstellungen, die für das jeweilige Erzeugnis eingesetzt werden. Ein stichprobenartiger, abschließender subjektiver Test prüft fahrzeugspezifische Modelle wie Audioanpassungen für das Fahrzeug.

#### 4.2.2.3 End-of-Line-Test

Die Serienfertigung schließt mit dem sogenannten End-of-Line-Test ab. Hier wird der SRK2 von der Fertigungs-Diagnosesoftware für ungültig erklärt und im „Revocation-Fuse-Register“ vermerkt. Ungültig zertifizierte Applikationen außerhalb der Zielapplikation mit dem SRK1 können von nun an nicht mehr ausgeführt werden. Merkmale, wie z. B. das Deaktivieren der JTAG-Schnittstelle für Debug-Optionen, Pin-Konfiguration oder andere Bootoptionen, stellen sicher, dass ein externer Bootvorgang von USB-Verbindungen oder Wechseldatenträgern nicht mehr durchgeführt werden kann. Das Navigationssystem hat somit im Bootvorgang einen bekannten Anfangszustand. Ein generierter Universal Unique Identifier (UUID) identifiziert das Gerät eindeutig. Diese Seriennummer wird im Secure Data Container (SDC) abgelegt und dem OEM zusätzlich in Analogie zu der Sicherheitsintegration aus Abschnitt 3.2.1.5 während des Fertigungsprozesses übermittelt.

Datenbanksysteme des OEM stellen nun Zertifikate für Zugangsberechtigungen wie z. B. Diagnosefunktionen bereit. Diese werden ebenfalls verschlüsselt im SDC abgelegt. Ein Berechtigungslevel auf Applikationsebene sorgt dafür, dass temporäre Funktionen ausgeführt werden können, die bewusst nicht in der Zielapplikation realisierbar sind. Im Anschluss werden die Geräte verpackt. Es folgt die Lieferung in ein Fertigungswerk zum OEM oder in ein Zwischenlager zum Vertrieb für Servicewerkstätten.

### 4.2.3 Umflashmaßnahmen im Werk

Im Vergleich zu anderen Domänen, wie dem Powertrain-Bereich, gibt es im Infotainment-Sektor bedeutend mehr Softwareaktualisierungen nach der Serienfertigung. Durch den Transport, z. B. über einen Seeweg, kann die Software am Bestimmungsort bereits nicht mehr aktuell sein. Abbildung 4.4 C) und D) stellt einen Prozessdurchlauf für ein Softwareupdate im Werk und den Anlernvorgang im Kfz bei der Endmontage im Werk des OEM dar.

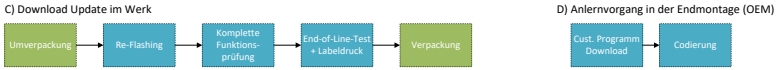


**Abbildung 4.4:** Updateprozess für den A-IVI

#### 4.2.3.1 Softwareupdate im Werk

Das Softwareupdate im Werk sieht vor, dass in Analogie zum Fertigungsprozess aus 4.2.2 auch nach dem eigentlichen Flashvorgang noch eine Funktionsprüfung und ein EOL-Test durchgeführt werden. Dieser ist dem elektrischen Test in der Serienfertigung angelehnt. Der wesentliche Unterschied liegt jedoch darin, dass hier der SRK2 für die Fertigungs- und Diagnosefunktionen nicht mehr gültig ist und nur noch die Zielapplikation mit dem SRK1 betrieben werden kann. Dadurch unterscheiden sich der Test und Updateprozess von der Serienfertigung. Um dennoch erweiterte Diagnose- und Installationsfunktionen nutzen zu können, wird ein in der

Applikation hinterlegter Autorisationslevel temporär freigeschaltet. Abbildung 4.5 zeigt einen schematischen Aufbau und macht deutlich, welche Diagnoseberechtigungen auf Applikationsebene freigeschaltet werden.

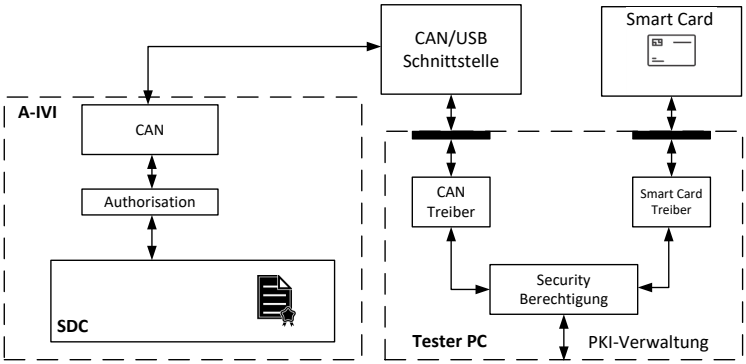


**Abbildung 4.5:** A-IVI-Diagnosefunktion

Das A-IVI wird im normalen Applikationsmodus betrieben. Hierzu ist ein Tester-PC via USB über ein CAN-Tool an das CAN-Interface der Head Unit angeschlossen. Der Tester-PC verfügt über eine Verbindung zur PKI-Verwaltung des OEM und außerdem über ein Smart-Card-Interface. Er baut eine Kommunikation über das CAN-Interface zur Head Unit auf. Hier muss sich der Tester authentifizieren, um ein höheres Autorisationslevel zu erhalten, damit zusätzliche Funktionen ausgeführt werden können. Dies geschieht über eine vom OEM „signierte Anfrage“ in Form einer „Level Change Request“-Datei. Der Anwender am Tester-PC verfügt über eine Smart Card, die mit einem individuellen Passwort zur Anfrage der signierten Datei vom OEM berechtigt. Der Tester-PC kann nun über den

PKI-Access die signierte Anfrage zur Head Unit stellen. Die Authentifizierung auf ein höheres Autorisationslevel wird durch einen Autorisation Level Daemon sichergestellt. Dieser überprüft die im SDC befindlichen Zertifikate, die in der Serienfertigung abgelegt wurden. Stimmt die Signatur der Anfrage mit den Zertifikaten überein, können nun diejenigen für Update- und Diagnosefunktionen ausgeführt werden.

#### 4.2.3.2 Anlernvorgang in der Endmontage des OEM

Im Fertigungswerk des OEM wird die Head Unit in das Kfz eingebaut und mit den zugehörigen Komponenten in der Domäne verheiratet. Neben Umgebungsbedingungen werden IDs anderer Geräte und die Fahrzeugidentifikation (FIN) eingetragen. In der Regel wird hier ein Fertigungsdiagnosetester verwendet. Die grundsätzliche Methodik der Authentifizierung bleibt analog zu der in Abbildung 4.5 dargestellten Form.

### 4.2.4 Umflashmaßnahmen im Feld

Nachdem die Head Unit nun im Fahrzeug verbaut worden ist und sich im Feld befindet, bedarf es einer Softwareaktualisierung.

Der OEM stellt das jeweilige Release für einen Download außerhalb des Werkes über ein Trust- und Management-Center in Analogie zu der Abbildung 3.4 zur Verfügung. Eine Software verwaltet über ein Berechtigungsverfahren die angestoßenen Updates. Softwareupdates wie Kartenmaterial oder Applikationsupdates werden in Installationspaketen vorbereitet. Definierte Softwarepakete werden in Abhängigkeit zu der bereits verwendeten

Software erstellt. Das Update kann physikalisch oder drahtlos over the Air erfolgen und ist in Abbildung 4.6 dargestellt.

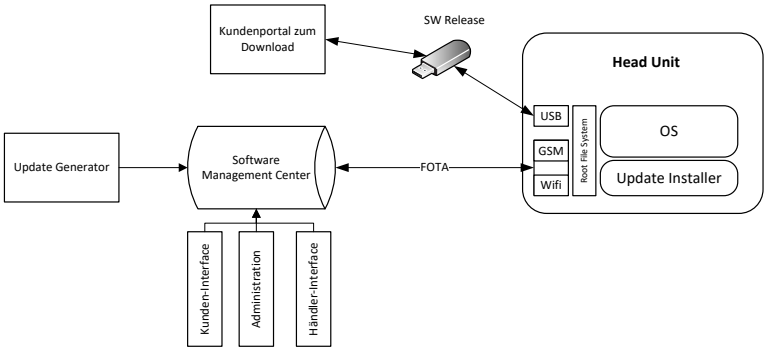


**Abbildung 4.6:** Softwareupdateverfahren Flash over the Air (FOTA) und mit USB-Stick

#### 4.2.4.1 Das Software Management Center

Der OEM stößt ein Update mittels eines Update-Generators an. Dies impliziert, dass im Vorfeld Freigabemaßnahmen und der Signierungsprozess aus Abschnitt 3.2.1.5 der Software vollzogen wurden. Nach einer Überprüfung der gerätespezifischen UUID der eingebauten Head Unit und der FIN des zugehörigen Kfz wird nun individuell eine signierte Installationsdatei erstellt. Hier wird von einer sogenannten „Manifest“-Datei gesprochen. Diese beinhaltet einen Abgleich der UUID in Form einer Index-Datei und den zu flashenden Softwarecontainer, abhängig von der vorhandenen Autorität (SRK1). Um Vertrieb und Roll-out der neuen Software zu gewährleisten,

gibt es hier mehrere Interfaces zu Kundenportalen oder Servicewerkstätten – auch für die Administration des Software Management Center.

Die grundsätzliche Methodik der Installation ist ebenfalls an die in Abbildung 4.5 dargestellte Funktionsweise angelehnt. Hier hat jedoch das Manifest, also die signierte Installationsdatei, einen für die Autorität gültigen Download-Request. Die Installation der neuen Software kann mit einem Diagnosetester oder auch in der Applikation der Head Unit selber angeregt werden.

#### 4.2.4.2 Flash over the Air

Bei einem Softwareupdate over the Air wird das zu installierende SW-Paket über viele kleine inkrementelle Softwarepakete zur Head Unit übertragen und im Flash zwischengespeichert. Erst wenn die Manifest-Datei komplett übertragen wurde, kann der Installationsprozess in der Applikation gestartet werden.

#### 4.2.4.3 Softwareupdate über ein physikalisches Medium

Ein Softwareupdate über ein physikalisches Medium, wie etwa einen USB-Stick, wird durch eine Anfrage, z. B. einer Servicewerkstatt, angeregt. Das Software Management Center generiert das Manifest und stellt es über ein Kundenportal zur Verfügung. In Abhängigkeit von der UUID des Gerätes wird der jeweils zu installierende signierte Softwarecontainer, der im Manifest hinterlegt ist, eingerichtet.

### 4.2.5 Der Wiederaufbereitungsprozess

Die Head Unit wurde nun in der Serie gefertigt. Neben der Bestückung der Leiterplatte, der Montage und Screening-Tests wurde der End-of-Line-Test des Zulieferers vorgenommen. Im weiteren Verlauf erfolgt der Einbau in das Kfz mit dem Anlernvorgang in der Serienfertigung des OEM. Mit aktueller Software überarbeitet, befindet sich die Head Unit nun im Feld. Regelmäßige Softwareupdates werden over the Air oder in einer Servicewerkstatt vorgenommen. Abbildung 4.7 zeigt ein klassisches Life-Cycle Model in Analogie zu dem im Abschnitt 3.1 vorgestellten Produktlebenszyklus, separiert zwischen den verschiedenen Ebenen der Serienfertigung, der Nutzung im Feld, der Anpassung in einer Servicewerkstatt und der Nachserienfertigung.

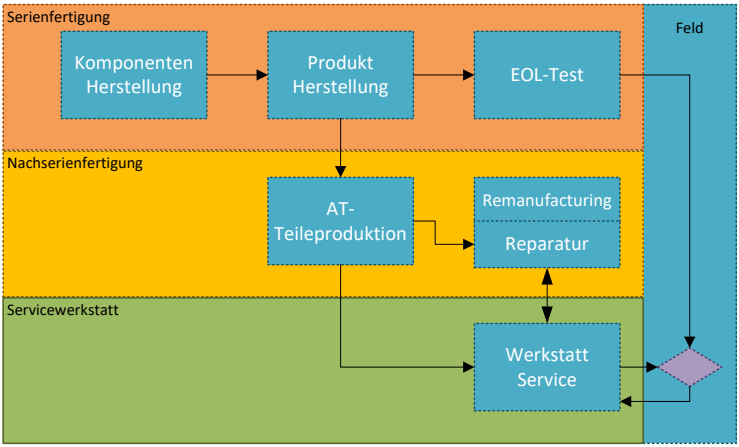


**Abbildung 4.7:** Life-Cycle Model einer Head Unit

Nun kann es jedoch vorkommen, dass die Unit nach einer gewissen Laufzeit einen Defekt aufweist, der die Applikation beeinträchtigt oder gar stört. In der Zeit der Nachserienversorgung hat der Endverbraucher das Recht, entweder ein äquivalentes Ersatzteil zu erhalten oder eine Reparatur vornehmen zu lassen. Hierzu werden Nachserienversorgungsstrategien angewandt, die im Abschnitt 3.4 vorgestellt wurden. Neben der Endbevorratung oder der Nachserienproduktion für Austauschteile (AT-Teileproduktion) kann ein Steuergerät individuell repariert oder auf Serienstandard wiederaufbereitet werden. Der Zulieferer versucht im Idealfall, eine teure Nachserienproduktion zu vermeiden.

Nach der Garantie entschließt sich der Fahrzeughalter, die Head Unit durch ein günstiges Reman-Gerät austauschen zu lassen. Die Servicewerkstatt sendet nun das Schadteil dem OEM und erhält ein wiederaufgearbeitetes Gerät zurück, dessen Softwarestand einer Neuware entspricht. Es folgen der Einbau und Anlernvorgang des Gerätes.

Vom OEM gesammelte Schadteile werden in einem zertifizierten Werk des Zulieferers wiederaufbereitet. Head Units werden final mit aktueller Firm- und Software sowie Kartenmaterial bereitgestellt, um sie als sogenannte „Reman-Ware“ zu vertreiben. Der ausführlich im Abschnitt 3.4.4 beschriebene Wiederaufbereitungsprozess beinhaltet im Wesentlichen ein Softwareupdate im Werk, eine elektrische Prüfung und Bedatung auf Fertigungsstandard. Dies impliziert unter Umständen auch eine vollständige Reparatur defekter Komponenten auf Leiterplattenebene. Basierend auf der Umflashmaßnahme im Werk in Abschnitt 4.2.3, wird der Prozess jedoch um einige Elemente aus der Serienfertigung erweitert, wie in Abbildung 4.8 dargestellt ist. Ein wesentlicher Unterschied gegenüber den bisher vorgestellten Umflashmaßnahmen liegt darin, dass noch nutzerbezogene Daten wie die FIN und UUID in dem Gerät hinterlegt sind. Diese

müssen ebenfalls gelöscht werden. Dies betrifft auch die beim OEM hinterlegten Zertifikate, die auf die UUID referenziert wurden.

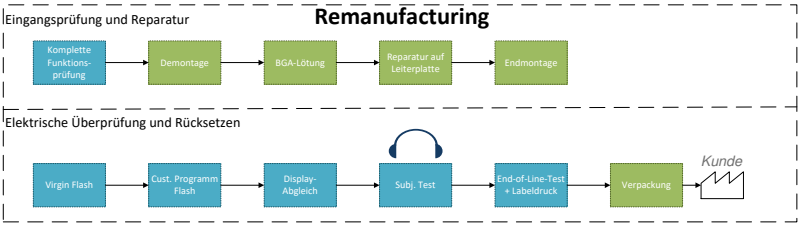


**Abbildung 4.8:** Reman-Prozess des A-IVI

Zunächst werden die vom OEM gelieferten Schadteile einer visuellen Inspektion unterworfen. Oftmals gibt es hier vereinbarte Ausschlusskriterien für eine weitere Bearbeitung, wie z. B. Wasserschäden oder direkt erkennbare „Tuningversuche“.

Als Nächstes durchläuft die Head Unit eine elektrische Funktionsprüfung. Diese lehnt sich stark an den Fertigungsprozess für eine Umflashmaßnahme im Werk an. Hier ist zu beachten, dass die Head Unit sich bereits in einem gesetzten „Secure-Device-Status“ befindet. Dies bedeutet, dass nur noch der SRK1 gültig ist und für erweiterte Funktionen eine Authentifizierung notwendig ist. Dies schließt die grundsätzliche Methodik der Installation anhand der in Abbildung 4.5 dargestellten Funktionsweise ein.

Mit Hilfe des Fehlerbildes wird nun der Reparaturprozess gestartet. Dabei kann es zur weiteren Aufbereitung nötig sein, die Leiterplatte vom Gehäuse und Display des Infotainment-Systems zu demontieren, um ausgefallene Komponenten auszutauschen. Dies können beispielsweise SMD-Kondensatoren sein, aber auch komplexe BGA-Bausteine, die eine besondere Löttechnik benötigen. Nach dem Komponententausch wird das Navigationsgerät wieder zusammengesetzt.

Im Anschluss folgt der sogenannte „Virgin-Flash“. Hierdurch werden alle Speicherbereiche des Infotainment-Systems zurückgesetzt. Typischerweise werden im Multimedia-Bereich Programmspeicherinhalte auf einem externen Flash gespeichert. Moderne Softwarepakete von Navigationssystemen liegen durch Kartenmaterial und Videodateien durchaus bei 64 GB Speicherplatz. Alternativ werden in der Wiederaufbereitung auch analog zu der Serienfertigung vorbeschriebene Speicherbausteine eingesetzt, um längere Übertragungszeiten im Flashprozess zu vermeiden. Nach dem Virgin-Flash werden analog zu der Serienfertigung zusätzliche Bedatungen vorgenommen sowie ein Displayabgleich und ein subjektiver Hörtest. Final folgt der sogenannte „End-of-Line-Test“.

### 4.2.6 Derzeitige Herausforderungen

Um das Problem der Abhängigkeit auf Typ-Teilenummer-Varianz zu mildern, gibt es verschiedene Lösungsvarianten. Der Wiederaufbereitungsprozess sieht derzeit vor, den i.MX6-Controller obligatorisch zu tauschen, um ein erneutes Flashen einer Software mit einer neuen Autorität zu gewährleisten. Dies ist jedoch eine vergleichsweise teure Variante, da neben der aufwendigen Lötung eines Ball-Grid-Array(BGA)-Bausteins auch das

Bauteil des i.MX6 kostenintensiv ist. Es bedarf eines hohen Engineering-Aufwandes, um einen BGA-Baustein zu entlöten.

Des Weiteren gibt es den Ansatz, dass vorbestückte Leiterplatten direkt als Ersatz verwendet werden. Hier ist die Leiterplatte mit einem bereits neuen i.MX6-Controller bestückt. Dieser Schritt verkürzt zwar die Prozesszeit im Gegensatz zum Entlötvorgang, dennoch ist eine langfristige Lagerung der Leiterplatte hinsichtlich Kosten und Requalifizierungsmaßnahmen sehr aufwendig.

Die aufgeführten Austauschprozesse beziehen sich auf die Annahme, dass der Prozessortyp für die gesamte Lebensdauer des Produktes verfügbar ist. Durch kürzer werdende Produktlebenszyklen, die bereits als Spannungsfeld in der Plattformentwicklung aus Abschnitt 3.2.3 bekannt sind, werden in der Halbleiterindustrie vermehrt Endbevorratungen durchgeführt, die sehr kostenintensiv sind und im Sinne der Revalidierung weitere Probleme der Lagerung mit sich bringen.

## 4.3 Handlungsbedarf

Dieser Abschnitt beschreibt die Einflüsse der Kernproblematik von Umflashvorgängen einer Software mit verschiedenen Autoritäten. Kontextbezogen zum Remanufacturing-Prozess aus 4.2.5 werden die Auswirkungen des irreversiblen Ablegens eines Schlüssels im Produkt für die Nachserien-Versorgungsstrategien dargestellt.

Im vorherigen Abschnitt wurden der Remanufacturing-Prozess für die A-IVI Head Unit und derzeitige Problematiken in der Wiederaufbereitung

erläutert. In diesem Prozess wird die Gegebenheit genutzt, dass hardwarekompatible Produkte, wie das A-IVI, auf einer anderen sogenannten „Typvariante“ geändert werden.

Eine neue Typvariante basiert auf einer kompatiblen Hardware mit einem neuen Softwarestand, die beispielsweise in einem anderen Automodell verwendet wird. Dies hat den Vorteil, dass anhand eines Forecast individuell und flexibel auf neue Abrufe reagiert werden kann, da hier verschiedene Softwarestände einen neuen Typ darstellen.

Im Zuge des Wiederaufbereitungskonzeptes kann sich durch eine neue Revision der Zielapplikation die Autorität ändern. Hardwarekompatible Head Units können infolge des irreversiblen Ablegens des Super Root Key (SRK) somit nicht mit signierter Software anderer Autoritäten geflasht werden.

In erster Linie wirkt sich das Problem dadurch aus, dass im Remanufacturing-Prozess die Wiederaufbereitungsquote verringert wird. Diese Quote richtet sich neben den Reparaturprozessen auch nach der jeweiligen Autorität der Software und nicht mehr ausschließlich nach Hardware-orientierten, referenzierten Grundtypen bei einem Umflashvorgang.

Die Flexibilität der Möglichkeit eines Umflashvorganges ist eine Schlüsselanforderung in der Wiederaufbereitung. Bei der höher werdenden Softwarevarianz zum Hardwarestand ist es entscheidend, diese Vielfalt nutzen zu können.

Langfristig unterstreichen weitere Problemfelder diese Anforderung an die Flexibilität. Die beschriebene Prämisse für die Materialverfügbarkeit durch kürzer werdende Produktlebenszyklen in der Halbleiterindustrie schränkt eine Ausführung von Endbevorratungen oder Vorproduktion von nicht beschriebenen Produkten (Rohlingen) mit ein.

Derzeitige Lösungskonzepte wie das beschriebene Remanufacturing haben Milderungseffekte auf Produkt- und Logistikkosten. Auch gibt es Einsparungen in der Materialverfügbarkeit elektronischer Komponenten. Grundsätzlich wird jedoch nur eine Problemverschiebung vorgenommen, die prozesstechnischen Anforderungen der Flexibilität über die Möglichkeit für den Umflashvorgang einer signierten Software werden langfristig nicht erreicht. Maßnahmen in der Fertigung und deren Einrichtungen sowie Prozesse der Wiederaufbereitung bilden eine Grenze für die Gestaltungsspielräume, um einen langfristigen Lösungsansatz realisieren zu können.

Ein weiterer Aspekt ist die Etablierung einer softwaretechnischen Lösung, die im existierenden Wiederaufbereitungsprozess Anwendung finden kann. So könnte der Software-Build-Prozess für die zu installierende Software erneut durchgeführt werden. Dabei wird im Signierungsabschnitt die Vertrauensquelle (privater Schlüssel) des Zielproduktes verwendet. Dies widerspricht jedoch dem Einsatz von signierter Software für ein vertrauenswürdiges SW-Management. Die Produkthaftung [105] besagt, dass der Hersteller des Endproduktes (OEM) gegenüber dem Verbraucher haftet. Unerwünschte Änderungen sollen durch Maßnahmen mit der Bereitstellung vertrauenswürdiger Hard- und Software verhindert werden. Eine Software für einen anderen Produktstand, sei es auch durch eine Hardwarekompatibilität des Produktes selbst, eröffnet die Möglichkeit, diese auch in der alten Produktumgebung einzusetzen.

Die Bedürfnisse aus der Nachserienversorgung für die Updatefähigkeit von Applikationen mit einer anderen Autorität bilden somit eine Anforderung an das Produktdesign.

Dadurch ist es notwendig, alle Akteure im Rahmen des Produktentstehungsprozesses mit einzubeziehen, um gemeinsame Strategien zu verfolgen. Für die Betrachtung eines Produktdesigns sind auch Bedingungen für

die Entwicklung neuer Geschäftsmodelle oder Kooperationsformen zur Umsetzung einer ressourceneffizienten Kreislaufwirtschaft zu implementieren. Insbesondere unter Nutzung der Möglichkeiten der Digitalisierung erhöht dies die Wiederverwendbarkeit eines Produktes.

Dies erfordert jedoch zusätzliche Funktionen und Schnittstellen für digitale Systeme zur Langzeitkompatibilität von Komponenten. Datenmanagement und zuverlässige Datenlöschung von Gebrauchtgeräten müssen ebenfalls in der Integration nachfolgender Wertschöpfungsstufen für ein Re-Use berücksichtigt werden. Somit müssen Strategien entwickelt werden, damit sich das Produkt zur Verwendung anderer und zusätzlicher Einsatzgebiete eignet, wie beispielsweise für Retrofit-Produkte.

Neben der Softwareentwicklung muss sich zwangsläufig auch mit Updates, Patches und dem Schutz der Software befasst werden. Gerade Berechtigungsmanagementkonzepte für den Produkteinsatz und zum Schutze des geistigen Eigentums des Herstellers werden vermehrt eine Rolle in Einsatz- und Ausrollkonzepten spielen. Diese Veränderung ergibt eine Serie von Nutzungs- und Zugangsrechten, die über die Zeit aktiviert, erneuert und aktualisiert werden. Diese Rechte steuern, welche Funktionalität des Erzeugnisses beim Kauf und der Aktivierung bereitgestellt wird.

## 4.4 Forschungsfrage

Durch den Handlungsbedarf kann nun die Zielsetzung der in diesem Kapitel aufgezeigten Problemstellungen erörtert werden. Spezifische Problemstellungen werden anhand eines erweiterten Life Cycle Model beschrieben.

Die Realisierung für die Updatefähigkeit von Applikationen mit einer anderen Autorität aus dem Beispiel des Remanufacturing-Prozesses stellt eine neue Anforderung an das Produktdesign der Head Unit. Die Motivation des Handlungsbedarfs entsprang der Kernproblematik, dass ein Update einer neuen Software in einem Steuergerät nicht nur hardwareseitig kompatibel sein muss. Gängige Methoden lassen nur eine kompatible Software in das Gerät flashen, wenn diese derselben Autorität entspricht.

Realisierungen der Wiederverwendbarkeit eines Produktes beginnen mit der Definition des Produktdesigns. Hier liegt der größte Ansatzpunkt darin, Möglichkeiten über den gesamten Lebensweg einer Ware und die spätere Kreislaufführung für Reparatur- und Upgradefähigkeit zu eröffnen. Das Design soll nicht nur die Kreislauffähigkeit verbessern, sondern insgesamt Produkte erzielen, die weniger Ressourcen in Anspruch nehmen. Eine Herausforderung besteht darin, dass aufgrund von Designanpassungen zusätzlicher Nutzen für die Kreislaufwirtschaft oft bei unterschiedlichen Akteuren entsteht (z. B. Signierungsprozesse der eingesetzten Software). Neue Designkonzepte zur Berücksichtigung der zusätzlichen Anforderungen der Kreislaufwirtschaft müssen wirtschaftlich tragfähig sein und sollen als Forschungsansatz integriert werden. Zudem wird eine Validierung des Designkonzeptes vorgenommen.

Konkret ergeben sich zwei Randbedingungen anhand der Forschungsfrage:

***Wie lässt sich eine Updatefähigkeit einer Software oder eines Datensatzes mit einer neuen Autorität für eine serviceorientierte Bereitstellung neuer Software-Komponenten realisieren, um die Wiederverwendbarkeit des Produktes sicherzustellen?***

### 4.4.1 Wiederverwendbarkeit des Produktes zur Etablierung einer zeitwertgerechten, ressourceneffizienten Kreislaufwirtschaft

Die im Abschnitt 4.2 gesetzte Forderung an das Produktdesign erfordert eine neue Betrachtung von derzeitigen Signierungs- und Authentifizierungsmethoden des Bootloaders und der Applikation. Dies schließt auch den Build-Prozess des Softwarecontainers mit ein. Ein Sicherheitskonzept soll sicherstellen, dass die Bedingungen unter dem gesetzten Handlungsfeld eingehalten werden.

In Abbildung 4.7 wurde das Life Cycle Model anhand einer Head Unit beschrieben. Nach der Produktherstellung mit dem zugehörigen End-of-Line-Test wird die Head Unit in das Fahrzeug verbaut. Im Rahmen der Verwendung im Feld gibt es Serviceintervalle, die in einer Werkstatt durchgeführt werden. Bei Defekten der Head Unit greifen Nachserienversorgungsstrategien wie das Remanufacturing oder die Individualreparatur.

Abbildung 4.9 beschreibt den aus dem Lösungskonzept abgeleiteten Punkt für den Re-Use-Aspekt einer ressourceneffizienten Kreislaufwirtschaft.

Die Erarbeitung neuer Lösungskonzepte für eine zeitwertgerechte, ressourceneffiziente Kreislaufwirtschaft stellt die letzte Anforderung an die Zielsetzung dar. Hierbei soll die Erweiterung existierender Geschäftsmodelle für den Einsatz der erarbeiteten Methodik betrachtet werden. Grundlage hierfür bildet die vorausgehende Aufgabenstellung für die Implementierung der Updatefähigkeit und zugehörigen Erneuerungen der Prozessdurchläufe.

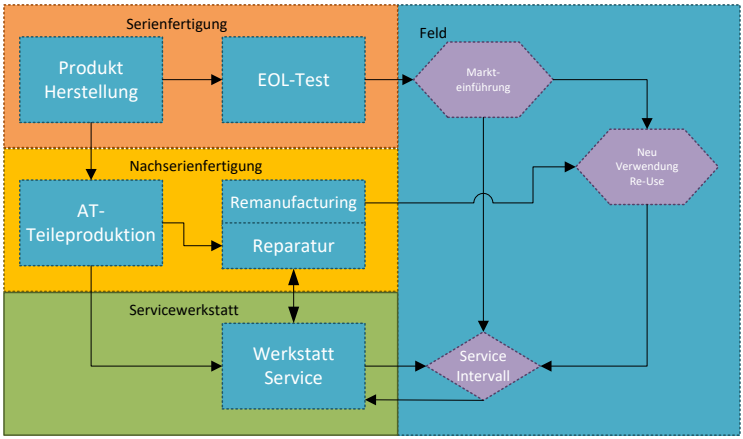


**Abbildung 4.9:** Ressourceneffiziente Kreislaufwirtschaft

### 4.4.2 Verbesserung und Erneuerungen von Updatefähigkeiten mit neuer, gültiger Autorität für eine Service-orientierte Bereitstellung neuer Softwarekomponenten

Der Serviceaspekt für die Implementierung neuer Softwarekomponenten im Rahmen der Nachserie ist ein Bestandteil moderner Servicestrategien. Neue Funktionalitäten lassen sich im Fahrzeug freischalten bzw. etablieren. Die Implementierung aktualisierter Softwarekomponenten knüpft an die Herausforderung der Updatefähigkeit mit neuer, gültiger Autorität an. Es bedarf eines Lösungsaspekts, um Softwarepakete von Drittparteien einzubinden.

## 4.5 Fazit

Die Benennung der Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer, gültiger Autorität zur Verbesserung des Produktlebenszyklus ist der zentrale Aspekt der Forschungsfrage. Im nächsten Kapitel wird schwerpunktmäßig der Lösungsaspekt der in Abschnitt 4.4 gestellten Fragestellungen behandelt.

# 5 Die Updatefähigkeit neu signierter Zielapplikationen

Dieses Kapitel beschreibt den Lösungsaspekt für die Realisierung der Updatefähigkeit von irreversibel geschützten Applikationen mit neuer Autorität. Im Mittelpunkt steht die Betrachtung der Fragestellung aus Kapitel 4: Die Benennung der Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer, gültiger Autorität zur Verbesserung des Produktlebenszyklus.

Der Lösungsansatz, eine Zielapplikation oder einen statischen Datensatz mit neuer Autorität betreiben zu können, besteht darin, zusätzliche Komponenten einzufügen. Hierbei handelt es sich um Autoritäten respektive öffentliche Schlüssel, die innerhalb einer jeweils laufenden Instanz für den Secure Boot weiterführender Codeblöcke/Applikationen oder statischer Datensätze verwendet werden. Die Separation bezieht sich darauf, dass die Autoritäten in einem eigenen Datenblock abgelegt werden. Durch diese Einführung lassen sich die Blöcke und somit die Autoritäten austauschen, ohne dass die laufende Applikation angepasst oder einem Update unterzogen werden muss. Die Methoden der Integritätsprüfungen und der Secure-Boot-Prozesses aus Abschnitt 2.3.1 sollen auf dem Stand der Technik und Wissenschaft aufbauen und gleichwertig umgesetzt werden.

Im Wesentlichen unterteilt sich dieses Kapitel in vier Bereiche:

Zunächst werden die Grundlagen des Secure-Boot-Vorganges aus Kapitel 2 aufgegriffen und Signierungs- sowie Validierungsprozesse der Softwareintegrationstests vorgestellt. Des Weiteren wird in Abschnitt 5.1 ein sich aus der Fragestellung ergebendes Basismodell hergeleitet. Dieses soll das betrachtete System und ihre einzelnen Komponenten beinhalten.

Anhand dieses Basismodells wird im zweiten Abschnitt unter 5.2 die Lösungsmethode entworfen. Zunächst wird die Kernanforderung der Updatefähigkeit einer Applikation mit neuer Autorität umgesetzt aus der Fragestellung in Abschnitt 4.4. Hierfür ist das Basismodell durch aktuelle Komponenten erweitert worden und die zusätzlich neu angefallenen Prozessstufen werden dargestellt.

Der Bedarf von Service-orientierten Bereitstellungen neuer Softwarekomponenten für nachfolgende Instanzen ist im dritten Abschnitt unter 5.3 beschrieben.

Hierzu wird im vierten Abschnitt unter 5.4 ein Bezug hergestellt, in Form eines erweiterten Modells angelegt, der die Service-orientierte Architektur in die Block-orientierte Lösungsmethodik einbindet. Insbesondere wird sich hier mit der Problemstellung der Speicherallokation beschäftigt und das Basismodell aus Abschnitt 5.2 angepasst.

## 5.1 Block-orientiertes Grundlagenmodell Secure Boot

Im folgenden Abschnitt wird grundlegend auf den im Abschnitt 2.3.1 beschriebenen Secure-Boot-Prozess eingegangen und ein Block-orientiertes Basismodell erstellt. Ein Secure-Boot-Prozess gewährleistet, dass sich die

auszuführende Applikation wie auch Codeblock und Image gegen die Hardware authentifizieren. Dies geschieht dadurch, dass der Code über einen vertrauenswürdigen Sicherheitsnachweis verfügt, der durch eine Vertrauensquelle verifiziert wird. Die Validierung der Vertrauensquelle durch die Hardware selbst bildet hierzu den ausgehenden Vertrauensanker, der im Abschnitt 2.2.2 erläutert wurde.

Gängige Praktik das signaturbasierte Verfahren aus Abschnitt 2.4.3, mit dem Gebrauch eines asymmetrischen Schlüsselpaares nach RSA [106], welche im Anhang unter A.2.2.2 erläutert ist. Der Secure-Boot-Prozess wird nachfolgend durch den Signierungs- und Validierungsprozess anhand einer allgemeingültigen Veranschaulichung dargestellt.

### 5.1.1 Signierungsprozess

Im Rahmen eines Softwareentstehungsprozesses startet der Signierungsprozess aus Abbildung 5.1, sobald das Image fertiggestellt ist. In der Regel wird ein projektzugehöriges (RSA) Schlüsselpaar generiert, das die Vertrauensquelle darstellt und in den Grundlagen in Kapitel 2 im Abschnitt A.3 beschrieben ist. Gängige Schlüssellängen betragen aktuell 2048 oder 4096 Bit. Langfristig wird jedoch die Kalkulation größerer Schlüssellängen Performance-Probleme verursachen [107]. Daher wird nach Zhang Peng et al. [108] ein ECC-Verfahren empfohlen, das mit kürzeren Schlüssellängen auskommt.

Aus dem zu signierenden Codeblock wird ein Hashwert berechnet. Gängige Verfahren sind hier der SHA2- und SHA3-Algorithmus [106]. Dieser Hashwert, der über die gleiche Modulo-Größe wie der öffentliche Schlüssel

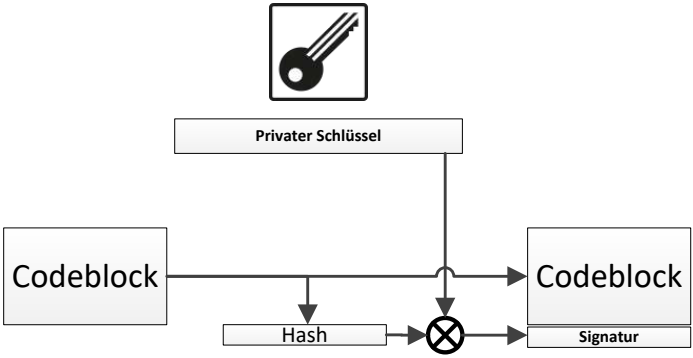


**Abbildung 5.1:** Signierungsprozess

verfügt, wird mit dem privaten Schlüssel verschlüsselt. Hierdurch entsteht die sogenannte Signatur, die den verschlüsselten Hashwert darstellt. Sie wird nun dem Codeblock als Epilog angefügt und im späteren Verlauf mit in das Gerät geflasht. Zudem ist es notwendig, den öffentlichen Schlüssel als Autorität gesichert im Gerät abzulegen.

### 5.1.2 Verifizierungsprozess

Der Verifizierungsprozess aus Unterabschnitt 2.4.3 bildet einen Hashwert aus dem auszuführenden Codeblock. Mit einem hinterlegten öffentlichen Schlüssel wird die Signatur entschlüsselt. Das sich ergebende Datum muss dem Hashwert des Codeblockes entsprechen.

Erst dann ist die Überprüfung abgeschlossen und sichergestellt, dass nicht manipuliert und der Autorität entsprochen wurde. Sollte sich der Programmcode geändert haben oder wie im Beispiel in Abbildung 5.2 mit einem anderen Schlüssel signiert worden sein, schlägt die Authentifizierung fehl. Der Codeblock würde für ungültig erklärt und nicht ausgeführt werden.

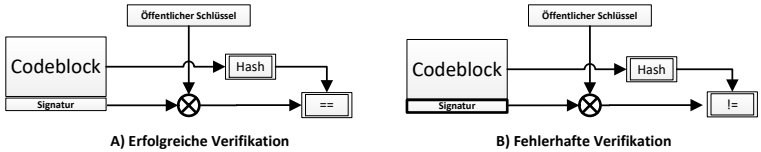


**Abbildung 5.2:** Validierungsprozess mit erfolgreichem Ergebnis A) und Ausfall B)

Um einen zusätzlichen Schutz der Informationsinhalte zu gewährleisten, kann der Datensatz bzw. das Image des Codeblockes verschlüsselt abgelegt werden. Äquivalente Funktionalitäten gibt es auch in anderen modernen Rechnerarchitekturen in [106], [102] und [109], wo gesonderte Adressbereiche in Verwendung sind.

### 5.1.3 Basismodell

Ein Secure-Boot-Prozess verläuft generell über mehrere Bootstufen. Somit etabliert sich eine Vertrauenskette, die nach Sabt et al. [28] wie folgt in Gleichung 5.1 dargestellt werden kann:

$$\begin{aligned} I_0 &= WAHR; \\ I_{i+1} &= I_i \ \wedge \ V_i(L_{i+1}) \end{aligned} \tag{5.1}$$

Die Integrität $I$ wird mit der Bootinstanz-Stufe i bezeichnet. $I_0$ stellt das irreversible Element dar, hier als öffentlicher Schlüssel Root benannt, das als vertrauenswürdig gilt. Im Grundlagenabschnitt 2.2.1 am Beispiel Fuse-Register können hier weitere interne Verifizierungsprozesse beinhaltet sein, um den gefusten öffentlichen Schlüssel selbst zu validieren. Im Folgenden wird jedoch das irreversible Element als gültig wahrgenommen, dessen Integrität gegeben ist.

$V_i$ ist die entsprechende Verifizierungsfunktion. $L_i$ bildet die Bootinstanzstufe ab. Letztendlich ist ohne eine Integrität des durch $I_0$ dargestellten irreversiblen Elementes eine anfängliche Integritätsprüfung wertlos. Deshalb ist der erste Start durch ein manipulationssicheres Hardwaremodul geschützt. Abbildung 5.3 verdeutlicht den allgemeinen Secure-Boot-Prozess.

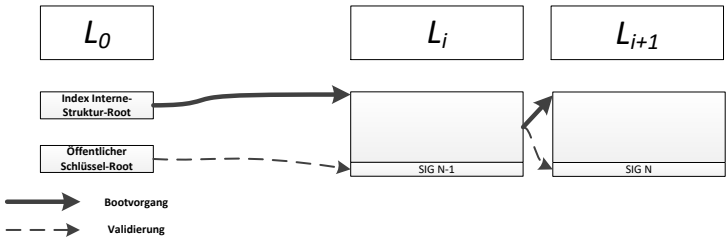


**Abbildung 5.3:** Allgemeiner Secure-Boot-Prozess

#### 5.1.3.1 Secure-Boot-Prozess mit zentralen Hardware-Security-Modulen

Anhand der am Leitbeispiel in Kapitel 4 im Abschnitt 4.2.1 erläuterten Rechnerachitektur des i.MX6 ergibt sich eine einheitliche Verifizierungsfunktion bei zentralisierten verorteten Hardware-Security-Modulen. Weiterhin gilt, dass die Integrität in Gleichung 5.2 der jeweils aktiven Instanz gegeben ist.

$$\begin{aligned} I_0 &= \ WAHR; \\ I_{i+1} &= I_i \ \wedge \ V_0(L_{i+1}) \end{aligned} \tag{5.2}$$

Die entsprechende Verifizierungsfunktion $V_0$ wird somit über jede Instanz $L_i$ angewandt und ist in Abbildung 5.4 dargestellt.

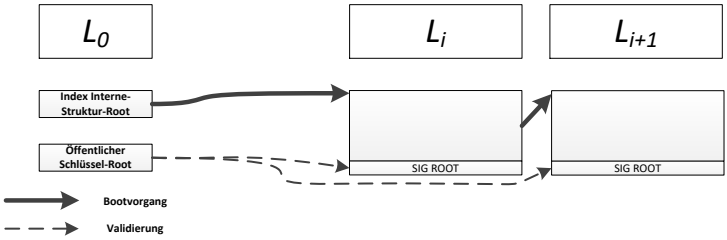


**Abbildung 5.4:** Allgemeiner Secure-Boot-Prozess bei zentralisierten Hardware-Security-Modulen

### 5.1.3.2 Instanz-orientiertes Basismodell

Um den Secure-Boot-Prozess nach Sabt et al. [28] in ein Basismodell zu überführen, wird eine Bootstufe als Instanz mit mehreren Blöcken definiert. Jede Instanz verfügt über einen sogenannten Root- und einen Datenblock. Der Root-Block besteht aus der laufenden Applikation, einem Index für die interne Datenstruktur und einem anderen mit zugehörigen öffentlichen Schlüssel zur nächsten Instanz $L_{i+1}$. Die Indizes enthalten zusätzlich IDs zur Softwareversionierungs- und Referenzüberprüfung. Der Datenblock beinhaltet statische Datensätze.

Abbildung 5.5 zeigt einen Bootprozess einer Instanz.

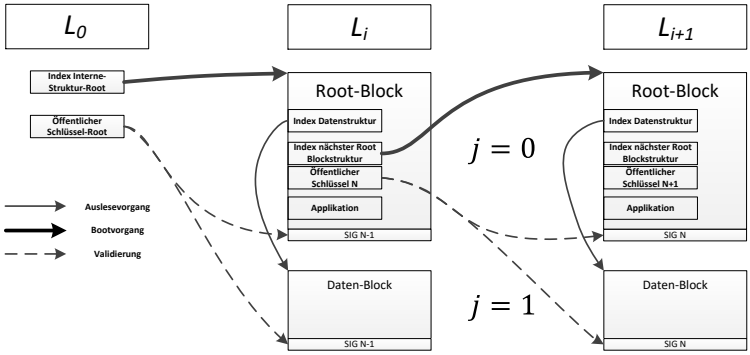


**Abbildung 5.5:** Bootvorgang der ersten Instanz

Der Software-Integrationstest (Secure Boot) für die erste Instanz startet von beispielsweise einem Boot-ROM ausgehend nach einem Initial-Reset.

Der Integrationstest beginnt mit dem den Aufruf eines Authentifizierungsbefehls eines Hardware-Security-Modules (HSM) in der Root-Instanz $L_0$. In dieser Sektion ist auch der der öffentliche Schlüssel $I_0$ des Gerätes untergebracht und wird hier als Root-Schlüssel bezeichnet. Ein Root-Index übergibt dem Boot-ROM bei Funktionsaufruf die Startadresse und Länge des Root-Blockes der ersten Instanz $L_i$. Nach erfolgreicher Authentifizierung durch das HSM wird die erste Instanz gestartet.

Die Besonderheit der Definition einer Instanz erzwingt die Einführung eines neuen Index, da innerhalb einer Instanz mehrere Datenblöcke enthalten sein können. Es besteht die Prämisse, dass die vorherige Instanz $L_i$ verifiziert und die Integrität $I_i$ gegeben ist. Die Instanz $L_i$ führt die Verifizierungsfunktion $V_i$ für die Instanz $L_{i+1}$ durch. Die Instanz $L_{i+1}$ setzte sich dahingehend aus dem Root-Block $L_{i+1,0}$ und den zugehörigen Datensätzen $L_{i+1,j}$ zusammen. Daher ist der Verifizierungsprozess $V_i$ ein Produkt der einzelnen gültigen, in der Instanz befindlichen Blöcke mit:

$$\begin{aligned} I_0 &= WAHR; \\ I_{i+1} &= I_i \ \wedge \prod_{j=0}^{j} V_i(L_{i+1,j}) \end{aligned} \tag{5.3}$$

Das hierin beschriebene Basismodell erfordert im Falle einer Änderung der Autorität einer Instanz weiterhin die Einbindung nicht betroffener Instanzen im Signierungsprozess. Sollte eine Instanz $L_{i+1}$ mit neuer Autorität aktualisiert werden, muss der Root-Block der vorherigen Instanz $L_i$ angepasst werden. Somit ergibt sich auch der Signierungsprozess der Autorität der Instanz $L_{i-1}$. Trotz der im Root-Block befindlichen Autorität steht die Forderung zur Verbesserung und Erneuerung von Updatefähigkeiten mit

neuer, gültiger Autorität für eine Service-orientierte Bereitstellung aktueller Softwarekomponenten noch aus.

## 5.2 Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer, gültiger Autorität zur Verbesserung des Produktlebenszyklus

Bereits in der frühen Phase einer Softwareentwicklung, die unter Abschnitt 3.2.2 beschrieben wurde, bedarf es einer Methode, mit der die Integrität der befindlichen Geräte zu überprüfen ist, insbesondere bei der Verwendung von in dieser Phase noch hohen Änderungen unterliegenden Softwareständen und Versionsinformationen eines Konfigurationssystems. Bei diesem Entwicklungszyklus ist eine Selbstvalidierung des Produktes respektive diejenige des Steuergerätes erforderlich, da jede Softwarekomponente, wie beispielsweise ein Bootloader oder eine Applikation, austauschbar sein könnte. Das impliziert schon in dieser frühen Produktlebenszyklusphase eine Anforderung, die verschiedenen Softwarekomponenten sowie deren Ursprung und Versionierung identifizieren zu können.

Saleker et al. [17] führen daher die Anforderungen der Identifizierung aller Komponenten innerhalb eines Systems an. Prinzipiell setzt sich ein Embedded System respektive Steuergerät aus mehreren Komponenten zusammen. Der Lösungsaspekt zur korrekten Selbstidentifikation dieser Komponenten liegt in der Generierung von Stücklistenobjekten, die mittels Metadaten Versionen, IDs, Zeitstempel und Pfadnamen der einzelnen Komponenten

beinhalten. Aus diesen drei Gruppen wird im System nun eine Hashberechnung vollzogen und somit der eindeutige Identifikationswert einer Instanz erzeugt. Dieser Identifikationswert wird als Bill of Material und im Folgenden als Index bezeichnet. Die Indizes werden zusammengefasst und zu einem Root-Index erstellt. Sollte diese Kombination als gut markiert werden, wird aus dem Root-Index ein validierter Index dupliziert. Sollte sich jetzt im Build-Prozess eine Änderung ergeben haben und sich somit der Root-Index modifizieren, wird dies sofort durch den validierten Index und den neu generierten Root-Index ersichtlich.

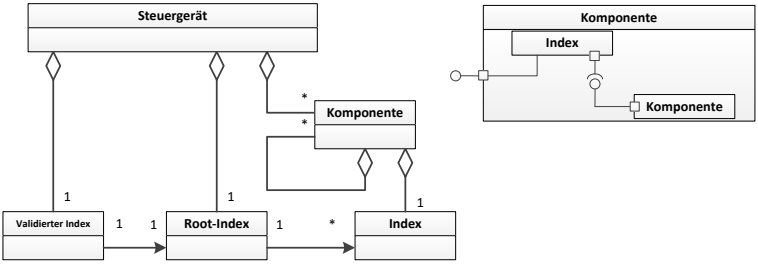


**Abbildung 5.6:** Komponentendarstellung eines Steuergerätes mit den Indizes nach [17]

Um die Anforderung einer Updatefähigkeit für eine Zielapplikation mit neuer, gültiger Autorität zu gewährleisten, muss jedoch der komplette Lebenszyklus für die Wiederverwendbarkeit des Produktes in Betrachtung gezogen werden, des Weiteren auch die Anforderung der Implementierung neuer Softwarekomponenten in der Verwendungszeit. Um dies zu erreichen, muss eine Trennung zwischen den Drittinstanzen $L_{i-1}$ und $L_{i+1}$

gegeben sein.

### 5.2.1 Lösungsansatz

Der Lösungsansatz, um eine Zielapplikation oder einen statischen Datensatz mit neuer Autorität betreiben zu können, besteht darin, zusätzliche Komponenten innerhalb einer Instanz einzufügen. Hierbei handelt es sich um dedizierte Autoritäten respektive öffentliche Schlüssel, die innerhalb einer jeweils laufenden Instanz für den Secure Boot weiterführender Codeblöcke/Applikationen oder statischer Datensätze verwendet werden. Die Separation bezieht sich darauf, dass die Autoritäten in einem eigenen Datenblock abgelegt werden. Durch diese Einführung lassen sich die Blöcke und somit die Autoritäten austauschen, ohne dass die laufende Applikation angepasst oder einem Update unterzogen werden muss.

Damit mehrere Softwarekomponenten innerhalb einer Instanz um eine Instanz erweitert, ersetzt oder aufs Neue übergeben werden können, müssen neben der Softwareintegrität auch die Positionen der Softwarekomponenten bekannt gemacht werden.

Ein Steuergerät besteht anhand des Leitmodells in Abschnitt 5.1 aus der Integritätsstufe respektive dem öffentlichen Schlüssel-Root, einem Index zum Verweis auf die erste Instanz und beliebig viele Instanzen selbst. Eine Instanz setzt sich aus mehreren Codeblöcken, die zusätzlich über einen weiteren öffentlichen Schlüssel verfügen, und einem zusätzlichen Index zusammen. Dieser Index wiederum kann innerhalb der Instanz als Schnittstelle auf einen weiteren Codeblock oder auf eine weitere Instanz verweisen.

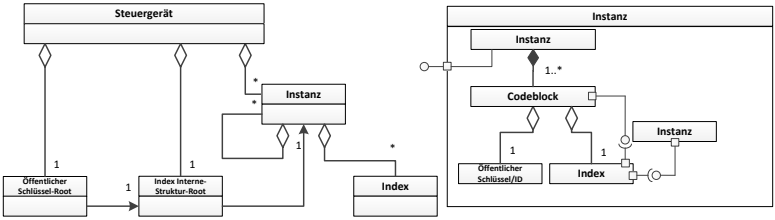


**Abbildung 5.7:** Komponentendarstellung eines Steuergerätes mit den Instanzen

### 5.2.2 Erweiterung des Basismodelles einer Instanz

Eine im Lösungsaspekt beschriebene Instanz besteht aus mehreren Konstellationen von Codeblöcken. Ein Block setzt sich aus verschiedenen Objekten zusammen. Im Grunde besteht ein Block aus einem Image, das mit einem statischen Programmcode bzw. Datensegmenten kombiniert sein kann. Da man von einem Secure-Boot-Vorgang ausgeht, sind diese Blöcke signiert. Eine Signatur ergibt sich aus dem verschlüsselten Hashwert eines Images und ist ebenfalls in einem Block enthalten.

Insgesamt gibt es vier verschiedene Arten eines Blockes, die in Abbildung 5.8 unter A bis D dargestellt sind. Der Root-Block A stellt den ersten aller Blöcke einer Instanz dar. Dieser Block beinhaltet eine laufende Applikation, die für die Instanz gültige Autorität und einen Index für die interne Struktur des nächsten Blockes.

Der zweite Block B wird als Interne-Struktur-Block bezeichnet und beinhaltet zwei Indizes. Der erste stellt Informationen über die in der Instanz befindliche Struktur der Schlüsselblöcke bereit. Mit diesem Index können alle nachfolgenden Schlüsselblöcke ausgelesen werden, die auf einen Root-Block einer neuen Instanz verweisen. Der zweite Index liefert die

Datenstruktur, wobei sämtliche für die Instanz benutzten Datenblöcke verwendet werden und zur Verfügung stehen. Eine Instanz besteht zwingend aus einem Root-Block und einem Interne-Struktur-Block.

Weiterführende Blöcke sind der Schlüsselblock C, der eigenständige öffentliche Schlüssel enthält, ferner einen Index für etwaige nächste Root- oder eigenständig signierte Datenblöcke. Zusätzlich sind hier auch IDs zur Softwareversionierung- und Referenzüberprüfung vorgesehen. Der in der Instanz zugehörige Daten-Block ist unter D dargestellt und beinhaltet statische Datensätze. Da es im Gegensatz zu dem Root- oder Interne-Struktur-Block mehrere Schlüssel- oder Datenblöcke geben kann, sind diese fortlaufend nummeriert.

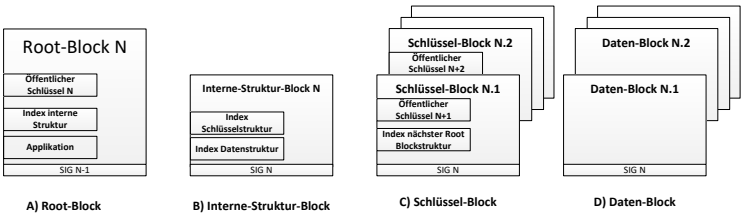


**Abbildung 5.8:** Definition der Blöcke einer Instanz

### 5.2.3 Bootvorgang

Der erweiterte Ablauf fokussiert ein Beispielszenario eines Secure-Boot-Vorganges. Hierbei sind die beschriebenen Komponenten aus Abbildung 5.8 dargestellt. Die Umgebung der laufenden Instanz $L_i$ besteht aus folgenden Komponenten:

- Root-Block $N$ mit enthaltener Autorität $N$
- Interne-Struktur-Block $N$
- Schlüssel-Block $N.1$ mit enthaltener Autorität $N + 1$
- Datensatz $N.1$

Weiterhin gilt, dass die Integrität des durch $I_0$ dargestellten irreversiblen Elementes nach wie vor gegeben ist und somit auch alle Elemente der Root-Instanz $L_0$:

$$I_0 = I_{0,j} = WAHR; \tag{5.4}$$

Nachdem die Integrität $I_i$ sichergestellt wurde, erfolgt die Verifizierung des Root-Blockes mit der Autorität $N$ $V_i, N(L_{i+1,\,0})$ aus der vorherigen Instanz und startet die Applikation des Root-Blockes $L_{i+1,\,0}$ und des darin enthaltenen öffentlichen Schlüssel $N + 1$ mit der Verifizierungsfunktion $V_{i+1,N+1}(L_{i+1,j})$ für die weiteren in der Instanz enthaltenen Blöcke.

$$I_{i+1} = I_i \ \wedge \ V_{i,N}(L_{i+1,j0}) \wedge \prod_{j=1}^{j} V_{i+1,N+1}(L_{i+1,j}) \tag{5.5}$$

Abbildung 5.9 zeigt den Bootvorgang, in dem drei Instanzen dargestellt sind. Instanz $L_i$ umfasst vereinfacht dargestellt einen Schlüsselblock,

die Instanz $L_{i+1}$ voll umfänglich einen einen Root-, Interne-Struktur-, Schlüssel- und Daten-Block. Die Instanz $L_{i+2}$ einen Root-Block.

Der Index Schlüssel-Blockes der Instanz $L_i$ validiert unter Zuhilfenahme des öffentlichen Schlüssels $N$ den Root-Block $N$ der Instanz $L_{i+1}$.

Der Index für die interne Struktur stellt im Root-Block $N$ zunächst die Informationen für den Interne-Struktur-Block $N$ bereit. Diese Metadaten sind physikalische Memory-Adressen, Blockgrößen sowie Versionierungen. Unter Zuhilfenahme des öffentlichen Schlüssels $N$ wird der Interne-Struktur-Block $N$ validiert und der darin enthaltene Index für die öffentliche Schlüssel- und Datenstruktur gelesen. Die beiden Indizes stellen ebenfalls Metadaten der in der Instanz enthaltenen Schlüssel- und Datenblöcke bereit.

Basierend aus den Informationen der Indizes wird der Schlüssel-Block $N.1$ für den öffentlichen Schlüssel $N + 1$ validiert und ausgelesen. Gleiches gilt für den Daten-Block $N.1$. Die Besonderheit eines Schlüssel-Blockes ist, dass hier ebenfalls ein weiterer öffentlicher Schlüssel $N + 2$ und eine Indexstruktur für den nächsten Root-Block $N + 1$ der Instanz $L_{i+2}$ bereitgestellt ist. Somit kann der nächste Root-Block validiert und in eine neue Instanz gewechselt werden.

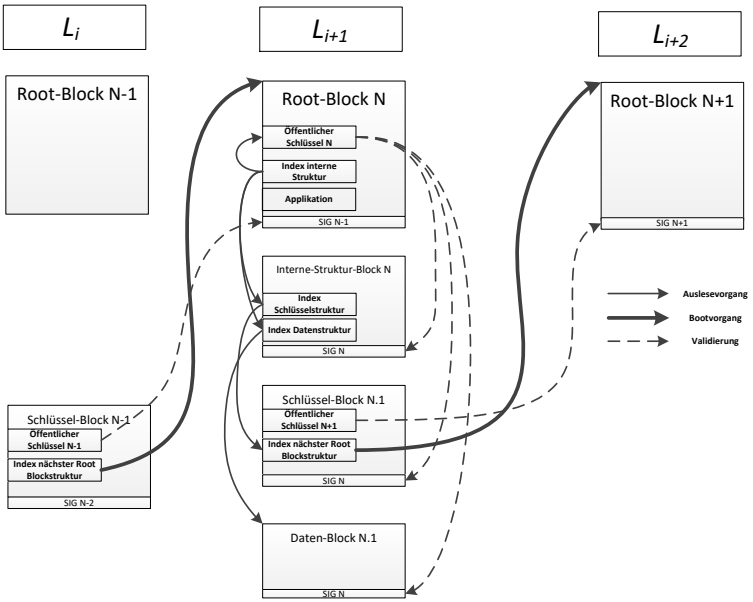


**Abbildung 5.9:** Bootabfolge einer Instanz

### 5.2.4 Datenblöcke dritter Parteien

Auch besteht die Möglichkeit, Datenblöcke mit einer eigenen Autorität innerhalb der aktiven Instanz zu verwenden. Dafür wird ebenfalls ein eigenständiger, zusätzlicher Schlüssel-Block, hier $N.1$ mit $L_{i,2}$ verwendet, um den öffentlichen Schlüssel $N+1$ zur Validierung bereitstellen zu können. Die Besonderheit liegt darin, dass im Index der Datenstruktur des Interne-Struktur-Blockes auf den Schlüssel-Block verwiesen wird. Es gilt somit für dieses Beispiel:

$$I_{i.3} = V_{i,N+1}(L_{i,3}) \tag{5.6}$$

Abbildung 5.10 zeigt einen Ausschnitt mit einem eigenständigen Schlüssel-Block $N.1$ und einem Daten-Block $N.1$ mit der Autorität $N+1$.

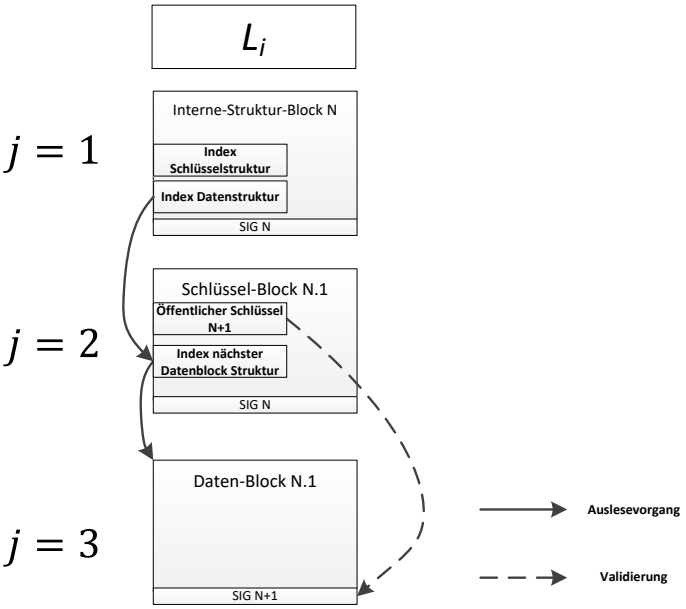


**Abbildung 5.10:** Betreiben eines Daten-Blockes mit eigener Autorität

### 5.2.5 Updatefähigkeit

Ändert sich beispielsweise im Rahmen eines Softwareupdates auch die Autorität $N + 1$ nach $N + 1'$ für den Root-Block $N + 1$, muss neben der Aktualisierung des Root-Blockes $N + 1$ auch in der vorherigen Instanz ein Update des gültigen Schlüssel-Blockes $N.1$ nach $N.1'$ vollzogen werden. Der daraus entstehende Vorteil ergibt sich dadurch, dass die Applikation der Instanz $L_i$ vom Updatevorgang nicht betroffen ist und das den Prozess des Updates behandeln kann. Abbildung 5.11 zeigt den Updatevorgang eines Root-Blockes mit Aktualisierung der Signaturquelle.

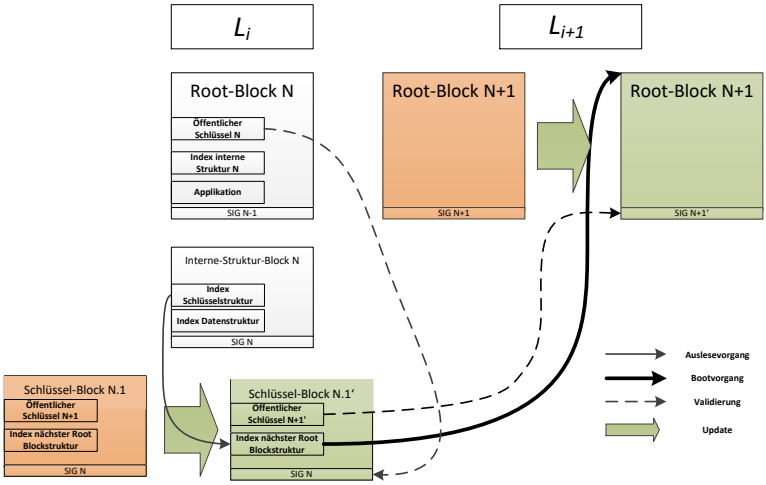


**Abbildung 5.11:** Updatevorgang eines Root-Blockes mit Aktualisierung der Signaturquelle

### 5.2.6 Berechtigungskonzepte

Ebenso können durch die partielle Updatefähigkeit Berechtigungskonzepte eingeführt werden. Als Beispiel verfügt Instanz $L_i$ über zwei Schlüssel-Blöcke $N.1$ und $N.2$ zu Folgeinstanzen $L_{i+1}$ und $L_{i+2}$, die zustandsgebunden aufgerufen werden. Nun soll temporär eine Nutzung der Folgeinstanz $L_{i+1}$ unterbunden werden. Stattdessen soll die Instanz $L_{i+2}$ zur Ausführung kommen. Um dieses Berechtigungskonzept zu ändern, ist in der laufenden Instanz $L_i$ nur der Interne-Struktur-Block $N$ auszutauschen, indem der Index für die Schlüsselstruktur angepasst wird. Updates für den Interne-Struktur-Block können mit einem niedrigen Aufwand, etwa über eine Funkschnittstelle, großflächig verteilt werden. Dies erfordert zudem nur eine geringe Bandbreite. Abbildung 5.12 zeigt den Updatevorgang des Interne-Struktur-Blockes $N$, wodurch der Index für die Schlüsselstruktur auf den Schlüssel-Block $N.2$ verweist.

Eine Instanz kann mit der Methodik aus Abbildung 5.12 mit dem Root-Block alle weiteren Prüfungen von den Blöcken eigenständig ausführen, sobald ihr eigenes Image durch die Prüfung ihrer Signatur als authentisch bestätigt wurde.

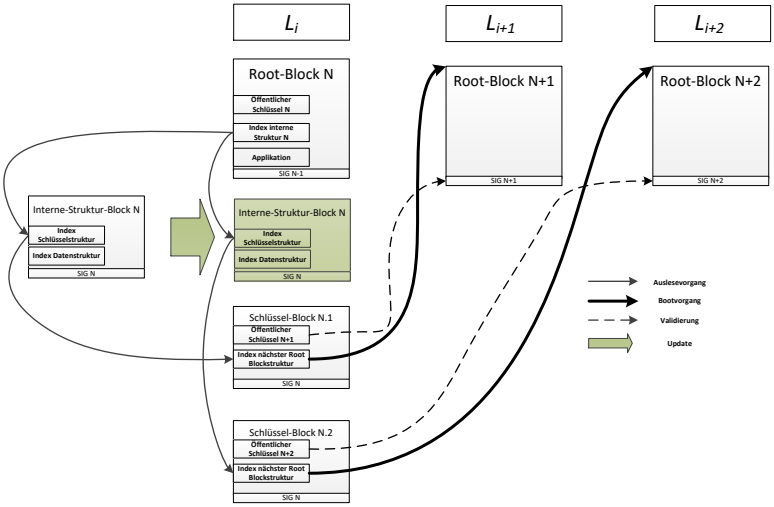


**Abbildung 5.12:** Updatevorgang des Interne-Struktur-Blockes N

## 5.3 Service-orientierte Bereitstellung neuer Softwarekomponenten

IT-Unterhaltungstechnologien respektive Konnektivitätsdienste bieten große Chancen, neue Anwendungen bereitstellen zu können, und werden vermehrt im Automobilsektor eingesetzt. Diese Technologien erfordern jedoch viel Anpassung, wodurch sich die existierenden Architekturen für die Automobilelektronik schnell angleichen. Insbesondere im Infotainment-Bereich bildet das Automobil ein verteiltes IT-System mit Cloud-Zugang. Im Fokus stehen jedoch Funktionsverbesserungen über Backendsysteme mit hoher Bandbreite im Hinblick auf Kartendienste, Medieninhalte sowie die umgebende Infrastruktur. [19] Speziell im Kontext autonomer Fahrzeuge müssen regelmäßig Softwareupdates durchgeführt werden. Dies ist

vor allem deshalb notwendig, um potentielle Sicherheitsrisiken und -fehler zu minimieren und neue Funktionen hinzufügen zu können. [110]

### 5.3.1 Domänenfusion

Derzeitige Updatestrategien beinhalten neben den klassischen Softwareupdates über einen angebundenen Diagnosetester auch Softwareupdates over the Air (OTA) mittels Connectivity-Modul, das über ein Backend kommuniziert. Das Fahrzeug muss zur Verwaltung eines Softwareupdates zu einem Händler gebracht oder geparkt und ausgeschaltet werden. In beiden Fällen führt dies zu erheblichen Ausfallzeiten und Unannehmlichkeiten für den Verbraucher. Künftige Herausforderungen, wie die Einführung des autonomen Fahrens, beinhalten Anforderungen, nach denen die Software nahtlos aktualisiert werden muss. Die Anforderungen machen es auch erforderlich, dass die E/E-Architektur innerhalb eines Fahrzeugs stärker vernetzt ist. Um die Sensorik miteinander zu fusionieren, muss unterschiedliche, dedizierte Hardware begrenzt werden. Dies schließt auch das Zusammenleben anwendungsspezifischer Betriebssysteme mit ein. [110]

Im Kapitel 2 wurde der allgemeingültige Aspekt eines Transformationsprozesses im automobilen Kontext anhand der Domänenfusion in Abschnitt 2.2.0.1 erläutert. Das Gateway übernimmt eine zentrale Funktion und weist somit die höchste Kommunikationslast des Fahrzeugs auf. Der Datenstrom im Rahmen eines Softwareupdates wird durch die Domänen-orientierte Struktur über mehrere zentrale Steuergeräte geleitet, um die Zielkomponente erreichen zu können. Diese Struktur führt zu einer weiter steigenden Komplexität, da viele Komponenten in die Fahrzeugarchitektur integriert sind. Daher bieten Zonen-basierte Architekturansätze eine Vereinfachung.

Abbildung 5.13 zeigt eine Zonenarchitektur, in der neben der funktionalen und ortsbezogenen Clusterung die einzelnen Komponenten durch ihre physische Platzierung im Fahrzeug unterteilt sind. [18]

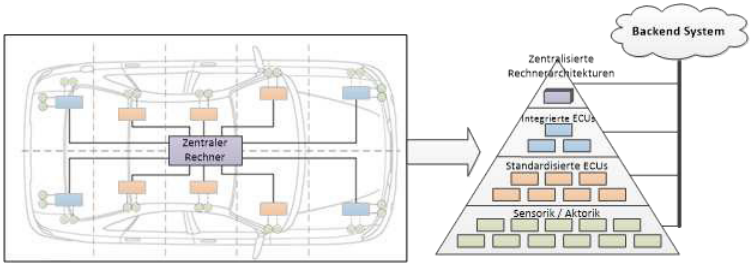


**Abbildung 5.13:** Zonenbasierte Architektur nach Brunner [18] und Christof Ebert [19]

Mit aufsteigender domänenübergreifender Kommunikation und lokaler Verarbeitung sind standardisierte und spezialisierte ECUs als Zonencontroller etabliert, die im Kern von der zentralen Rechnereinheit eine Weiterleitungs-, jedoch keine Bearbeitungsaufgabe vollziehen. Mit diesem Ansatz kann auch ein Backend als eine Verarbeitungseinheit realisiert werden. Diese Unterstützung bietet eine hohe Flexibilität für die Integration von Funktionen durch das Hinzufügen von Softwarekomponenten in der zentralen Rechnereinheit oder im Backendsystem.

### 5.3.2 Service-orientierte Architektur

Der Ansatz einer zentralisierten Rechnereinheit erfordert eine hohe Integration von Softwarearchitekturen in der Automobilelektronik. Hierbei sind dynamische Betriebssysteme für die Automobilbranche erforderlich. So ermöglicht die Adaptive-AUTOSAR-Architektur [25] neue Wege zu einer strukturieren Gestaltung künftiger E/E-Architekturen.

Software-gesteuerte Funktionen in einem Automobil werden als sicherheitskritisch eingestuft. Bei fortlaufender Komplexität der beteiligten Hardwarekomponenten und -topologien wird vor allem der Fokus in Bezug auf die Qualität der Architekturspezifikationen gelegt. [111]

Mit formalen Methoden werden Service-orientierte Ansätze verfolgt, die die Architektur mit Eigenschaften wie dem Systemlebenszyklus, der Verständlichkeit, Erweiterbarkeit, Wartbarkeit und Wiederverwendbarkeit insbesondere im Hinblick auf durchgängige Software und Systemtechnik beschreibt.

#### 5.3.2.1 Layer-orientierter Ansatz

Für die Erläuterung einer Service-orientierten Architektur kommt ein Layer-orientierter Ansatz zum Tragen, der Software-Stacks und physische Hardwareelemente berücksichtigt. Nach [4] und [112] ist ein System mittels dreier Schichten beschrieben, separiert und in Abbildung 5.14 dargestellt. Im Kern werden hier Softwarekomponenten hinsichtlich der Berechtigungskonzepte (Berechtigungsstufen) voneinander getrennt.

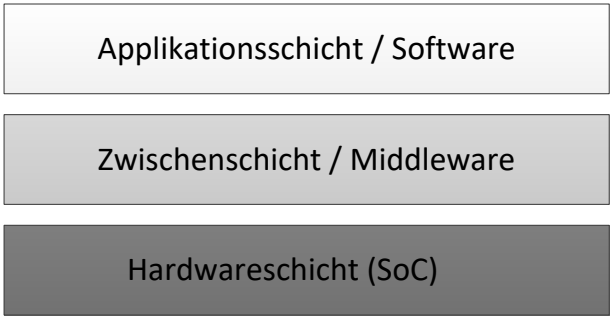


**Abbildung 5.14:** Layer-orientierter Ansatz nach [4]

Die Applikationsschicht mit ihren Softwarekomponenten enthält alle funktionalen Anwendungen eines Steuergerätes, wie beispielsweise eine Applikation oder statische Datensätze. Die Zwischenschicht wiederum beschreibt Service-orientierte Funktionalitäten, wie z. B. Betriebssysteme. Die Hardwareschicht bildet die Konfigurationsschnittstelle des SoC und wird dafür verwendet, um Hardwarekonfigurationen, Berechtigungen und Funktionen der Speicherzugriffe anzuwenden.

Grundsätzlich ist es somit möglich, bestimmte Aspekte innerhalb mehrerer Schichten voneinander zu unterscheiden, auch um beispielsweise domänenübergreifende Berechtigungskonzepte dritter Parteien anzugeben. Abbildung 5.15 zeigt ein Beispiel eines Berechtigungskonzeptes eines Aftermarket-Servicekonzeptes für den Zugriff Dritter auf im Fahrzeug

generierte Daten. [20]

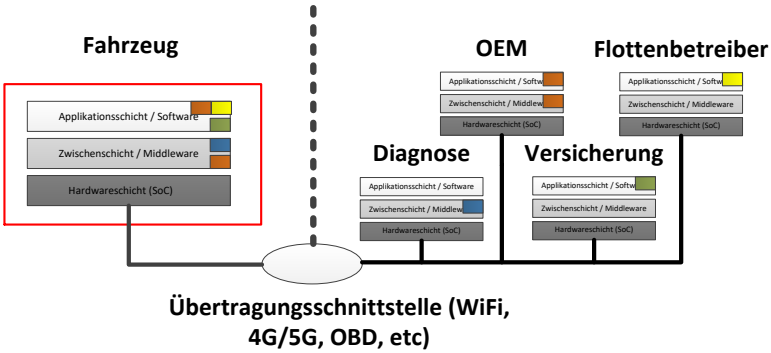


**Abbildung 5.15:** Berechtigungskonzept eines Aftermarket-Service-Konzeptes im Layer-orientierten Modell nach [20]

Vorqualifizierte Softwarekomponenten sind allgemein als IP-Blöcke (IP = Intellectual Property) bekannt. Diese Blöcke oder Kerne sind üblicherweise im SoC-Layout vertreten, kommen jedoch auch für Verwaltungsschnittstellen oder Datensätze, wie beispielsweise trainierte Gewichte, zum Einsatz.

## 5.4 Memory Mapping im Block-orientierten Kontext

Für die Service-orientierte Bereitstellung neuer Softwarekomponenten dritter Parteien und somit auch von Blöcken mit verschiedenen Autoritäten soll nun der Layer-orientierte Ansatz in das Block-orientierte Modell überführt werden.

Zunächst wird in Abbildung 5.16 ein Modell für das Memory Mapping vorgestellt. Diese überführt die Layer-orientierte Darstellung von Softwarekomponenten dritter Parteien in das Basis-Blockmodell aus Abschnitt 5.2.

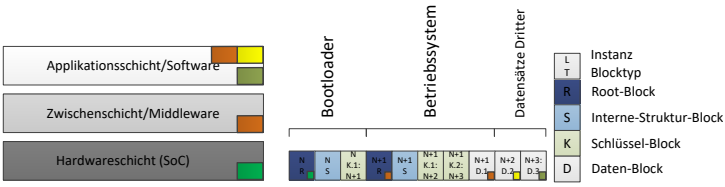


**Abbildung 5.16:** Darstellung eines horizontalen Mappings von drei Instanzen

In diesem Beispiel besteht das System aus einem Bootloader und einem Betriebssystem, die in separaten Instanzen auftreten. Eine Instanz verfügt über einen Root-Block $R$ und den Interne-Struktur-Block $S$, optional über einen Schlüsselblock $K$ und einen Datenblock $D$.

Der Bootloader, hier Instanz $L_i$, lässt sich beschreiben durch den Root-Block $\overset{N}{R}$, den Interne-Struktur-Block $\overset{N}{S}$, den Schlüssel-Block $\overset{N}{K.1 : N+1}$ zum Booten des Betriebssystems, die in der Instanz $L_{i+1}$ verortet ist.

Das Betriebssystem verfügt über mehrere Softwarekomponenten in Form des Daten-Blockes $\overset{N+1}{D.1}$ und Datenblöcke dritter Parteien $\overset{N+2}{D.2}, \overset{N+3}{D.3}:$ mit zugehörigem Schlüssel-Block $\overset{N+1}{K.1 : N+2}$ und $\overset{N+1}{K.2 : N+3}$ zum Bereitstellen des eigenen öffentlichen Schlüssels.

Für Softwareaktualisierungen muss auch sichergestellt werden, dass keine anderen, angrenzenden Blöcke im Sinne des Memory Mapping verschoben oder gar überschrieben werden. Anhand von zwei Leitbeispielen im folgenden Abschnitt soll die Problematik zur Speicherallokation neuer Komponenten und Instanzen dargestellt werden.

Anschließend wird anhand des Basismodells aus Abschnitt 5.2 eine erweiterte Version bereitgestellt.

### 5.4.1 Problemstellung der Speicherallokation am Beispiel eines alternativen Bootloaders

Im folgenden Abschnitt soll die Problemstellung der Speicherallokation verdeutlicht werden. Eine Instanz kann ihre eigene Struktur oder eine Folgeinstanz aktualisieren. Die jeweiligen Applikationen respektive Softwareblöcke selbst sind nach 2.5.1 und am Leitbeispiel der i.MX6 Plattform

in Abschnitt 4.2.1.2 jedoch im Build-Prozess der Software einem festen System-Adressbereich zugeordnet.

Der Root-Block bezieht den nächststehenden Interne-Struktur-Block über einen Index, der Interne-Struktur-Block wiederum über die Indizes der jeweiligen nachfolgenden Schlüssel- oder Datenblöcke. Im jeweiligen Schlüsselblock befindet sich folglich ein Index zur nachstehenden Instanz oder zu einem Daten-Block.

Die System-Adressbereiche sind in den Indizes statisch hinterlegt, womit die Position und somit auch die Speicherallokation im Vorfeld definiert sein müssen. Die fortlaufende Hierarchie ergibt sich demnach durch die Verkettung der in den Index-Strukturen angegebenen Adressbereichen. Damit ist allen Instanzen eine zugehörige Struktur vorgeschrieben.

In dem folgenden Beispiel soll die Problematik der Speicherallokation im Rahmen eines Softwareupdates verdeutlicht werden indem ein Bootloader durch einen alternativen Bootloader erweitert wird.

Das Grundmodell fußt auf dem in Abschnitt 5.4 dargestellten Memory Mapping in Abbildung 5.16. Die Instanz $L_i$ und $L_{i+1}$ beschreiben einen zweistufigen Bootloader. Instanz $L_{i+2}$ soll das Betriebssystem bilden.

Nun soll ein alternativer Bootloader 2 als Instanz $L_{i+3}$ eingeführt werden. Das Betriebssystem der Instanz $L_{i+2}$ kann sowohl über den Bootloader 1 mit der Instanz $L_{i+1}$ als auch über den Bootloader 2 mit der Instanz $L_{i+3}$ aufgerufen werden. Dazu werden die beiden Datenblöcke $\overset{N+1}{D.1}$ und $\overset{N+1}{D.2}$ gelöscht, der Interne-Struktur-Block $\overset{N}{S}$ nach $\overset{N}{S\prime}$ aktualisiert. Die

Instanz $L_{i+1}$ zum Bootloader 1 bekommt zusätzlich einen eigenen Daten-Block $\overset{N+1}{D.1}$, wobei der existierende Interne-Struktur-Block $\overset{N+1}{S}$ nach $\overset{N+1}{S\prime}$ aktualisiert wird. Die neue Instanz $L_{i+3}$ des Bootloaders 2 verfügt analog zur Instanz $L_{i+1}$ über einen Root-Block $\overset{N+3}{R}$, Interne-Struktur-Block $\overset{N+3}{S}$, den Schlüssel-Block $\overset{N+3}{K.1:N+2}$ zum Booten der Instanz $L_{i+2}$ und dem Daten-Block $\overset{N+3}{D.1}$.

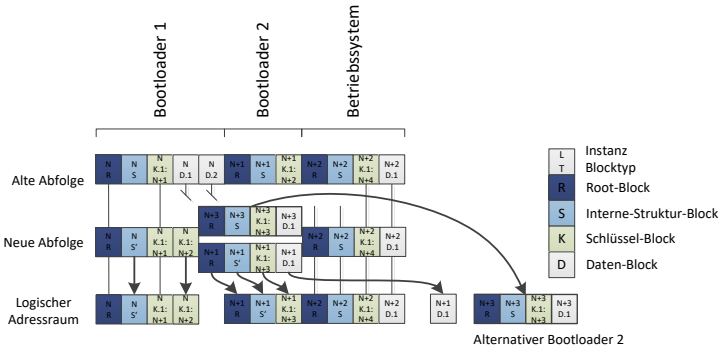


**Abbildung 5.17:** Veranschaulichung eines Memory Mappings im Rahmen eines Updatevorganges, hier: Bootloader

Das Kernproblem ergibt sich dadurch, dass jede Instanz seitens ihrer Blockstruktur, beginnend mit dem Root-Block mit beinhalteter Applikation, auf einer vordefinierten System-Adresse festgelegt ist. In diesem Beispiel wird die Instanz $L_{i+2}$ wahlweise über $L_{i+1}$ oder $L_{i+3}$ gebootet. Das Bedeutet, dass die Bootloader zwar im selben System-Adressbereich ausgeführt werden können, ggf. einander unbekannt sind, jedoch zwangsläufig in verschiedenen Speicher-Adressbereichen abgelegt sein müssen.

Im Zweifel entsprechen die Instanzen einer anderen Signaturquelle respektive einer anderen Partei, die nicht in den Build-Prozess der nachfolgenden Instanz involviert ist. Hieraus entsteht die zentrale Frage, inwieweit bei einem Softwareupdate keine Instanz durch eine andere in ihrem Bereich verletzt werden kann. Ein Lösungsaspekt liegt in der Befähigung einer Instanz, der jeweils nachfolgenden einen Adressbereich zuzuordnen. Dadurch muss eine Befähigung geschaffen werden, die es dem Root-Block ermöglicht, eine physische Zwischenadresse einzuführen oder gar eine virtuelle Adressierung, die im Rahmen seiner Instanz, aber auch der Folgeinstanz Platzierungen im Speicher-Adressbereich vollziehen kann.

### 5.4.2 Anpassung des Basismodelles zur variablen Speicherallokation der Softwarekomponenten

Eine variable Anpassung der internen Struktur setzt voraus, dass die verschiedenen Indizes während eines Softwareupdates und somit auch während der Laufzeit verändert werden müssen. Anhand des überarbeiteten Leitmodells erweist es sich als problematisch, wenn die Indizes schon vor dem Signierungsprozess eines Blockes bekannt sein müssen. Somit muss das Basismodell aus Abschnitt 5.2 in dem Sinne angepasst werden, dass

die Indizes der jeweiligen Blöcke unabhängig vom Signierungsprozess und außerhalb des Bereichs liegen, worüber eine Signatur gebildet wird. Abbildung 5.18 zeigt die Neudefinition der Blöcke einer Instanz.

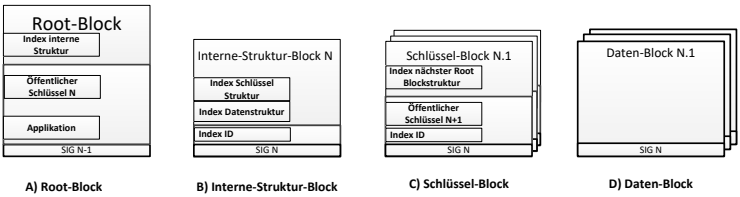


**Abbildung 5.18:** Neudefinition der Blöcke einer Instanz

Der Kernaspekt ist die Separation der Indizes eines Blockes. Weiterhin gilt der Signierungsprozess für öffentliche Schlüssel, insbesondere für den Root-Block und Schlüssel-Block. Im Falle notwendiger Informationen hinsichtlich der Struktur- und Schlüssel-Blöcke werden geforderte IDs zur Softwareversionierungs- und Referenzüberprüfung in den Signierungsprozess mit aufgenommen. Es gilt weiterhin die in 5.2.3 gesetzte Secure-Boot-Folge.

Somit ist die Grundlage gegeben, dass der Root-Block einer Instanz über seinen Interne-Struktur-Block eine eigene Partitionierung vollziehen kann, weiterführend auch zum Handling einer neuen Softwarekomponente respektive einer Folgeinstanz samt logischem Adressbereich. Abbildung 5.19 zeigt den Updatevorgang eines Root-Blockes mit einer Vererbung der Adressierung.

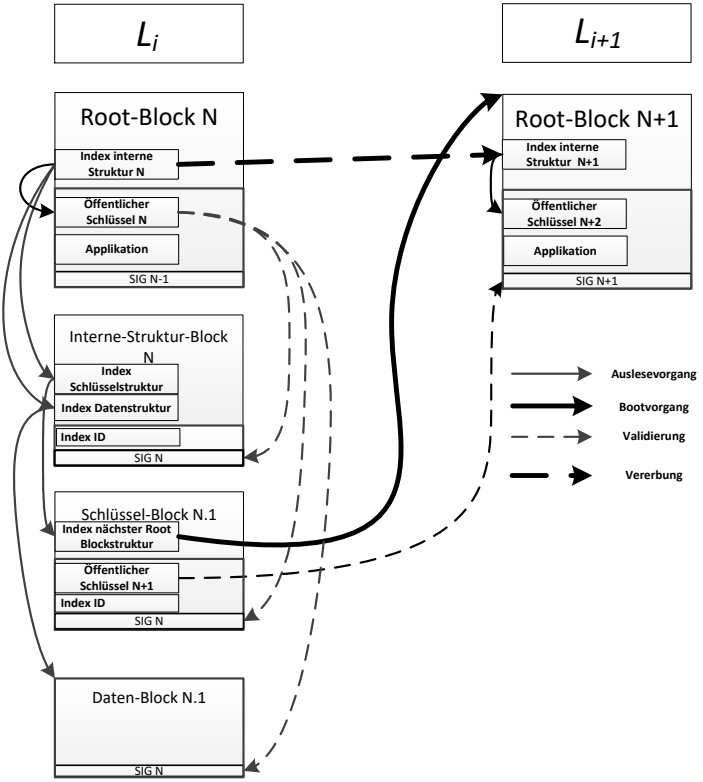


**Abbildung 5.19:** Updatevorgang eines Root-Blockes mit Vererbung der Adressierung

Durch die Vererbung eines Adressbereiches eines Root-Blockes kann die Folgeinstanz darüber eine eigene Partitionierung betreiben. Dies geschieht über eine hierarchische Vererbung, wonach jede Instanz ihr Mapping und das der vorherigen vererbt. Diese Vererbung enthält auch eine rückführende Möglichkeit, Parallelinstanzen zu koordinieren. Abbildung 5.20 zeigt eine

hierarchische Baumstruktur des Systems.

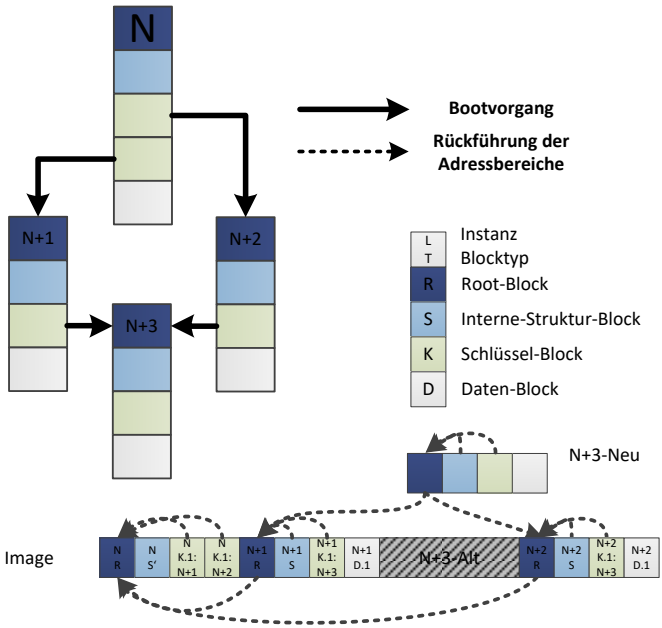


**Abbildung 5.20:** Parallelinstanzen

In diesem Beispiel ist $L_i$ die erste Instanz. Von $L_i$ ausgehend wird über Instanz $L_{i+1}$ auf Instanz $L_{i+3}$ gebootet. Möglich wäre auch die Bootreihenfolge ausgehend von der ersten Instanz $L_i$ über $L_{i+2}$ mit $L_{i+3}$.

Soll nun Instanz $L_{i+3}$ aktualisiert werden, kann über die Indizes der jeweiligen Instanzen ein neuer Adressbereich definiert werden.

Dazu muss anhand der gegebenen Baumstruktur die erste Instanz Kenntnis über den gesamten Adressbereich haben. Es muss somit ein Synchronisationsprozess angestoßen werden, der den jeweiligen Bereich der übergeordneten Instanz rückmeldet, damit eine korrekte Verteilung bei Parallelinstanzen eingefügt werden kann.

## 5.5 Fazit

Dieses Kapitel gewährt einen Lösungsaspekt zur Benennung der Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer, gültiger Autorität zur Verbesserung des Produktlebenszyklus.

Der wesentliche Aspekt liegt darin, dass durch den Verheiratungsprozess der Autorität ein gleichwertiger Aspekt zum Schutze der Software in der Integrität gegeben ist. Der öffentliche Schlüssel für einen signierten Daten- oder einen Root-Block ist selbst in einem eigenständigen Block signiert abgelegt.

Mit der neuen Modellierung wurde eine Grundlage dafür geschaffen, dass unter Beibehaltung des Secure-Boot-Prozesses mit einem irreversiblen Element auch neue Autoritäten eingebunden werden können und somit auch keine unbeteiligte dritte Partei am Signierungsprozess beteiligt werden muss. Des Weiteren sind Funktionalitäten für Berechtigungskonzepte und Updatemöglichkeiten, darüber hinaus auch neue Konzepte für die Softwaremodalität geschaffen worden. Ferner wurde auch eine Überleitung im

Sinne einer Service-orientierten Architektur hin zu dem Block-orientierten Modell verfolgt.

Speziell für die Service-orientierte Bereitstellung neuer Softwarekomponenten nachfolgender Instanzen muss eine genaue Betrachtung erfolgen. Der Kernaspekt, der Schutz der Datenintegrität zu schützen, ist mit dem zweiten, überarbeiteten Basismodell zugunsten der flexiblen Änderung der Adressbereiche statischer Daten aufgehoben worden. Die Indizes stehen nicht mehr in einem Bereich, der beim Software-Integritätstest inbegriffen ist. Daher muss eine genaue Betrachtung im Sinne einer Risikoanalyse erfolgen, um den Nachweis zu erbringen, das gesetzte Security-Niveau beizubehalten, wenn nicht sogar zu erhöhen.

Mit einer Bedrohungs- und Angriffsanalyse sollen Gefährdungen durch die zerlegten Systemelemente identifiziert werden. Die daraus resultierenden Gegenmaßnahmen sind erforderlich, um eventuelle Schwachstellen im System und in der Modellierung zu finden.

# 6 Sicherheitskonzept

Um sicherzustellen, dass die vorangestellten Lösungsaspekte im Kapitel 5 die in Kapitel 4 gesetzten Ziele erreichen, müssen diese nachweislich evaluiert werden.

Im Kern soll die Frage beantwortet werden, ob die angewandte Methodik aus Kapitel 5 ein erhöhtes Sicherheitsrisiko für den Schutz der Integrität eingesetzter Softwarekomponenten darstellt.

Hierzu wird im folgenden Abschnitt anhand des Leitbeispiels eine Angriffs- und Risikoanalyse (kurz T&R für Threat & Risk) erstellt. Die T&R-Analyse wird gemäß den wesentlichen Richtlinien der modellbasierten Entwicklung funktional sicherer Hardware nach ISO 26262 [59] und Automotive-Security nach ISO 21434 [64] durchgeführt, die im Abschnitt 3.2.1.2 erläutert wurde.

Methodiken zur Evaluation von IT-Sicherheitstechniken sind an ISO 18045 [113], ISO 27001 [66] und Sicherheitstechnik für Straßenfahrzeuge nach ISO 21434 [64] und der SAE-J3061 (Automotive Security Requirement Engineering) [65] angelehnt. [114]

Die Kernbetrachtung dieses Sicherheitskonzeptes liegt in der Evaluierung des Block-orientierten Modells zur Separation der Zielapplikation zum

Steuergerät nach Kapitel 5. Das Sicherheitskonzept besteht im Wesentlichen aus der hierin beschriebenen Risiko- und Bedrohungsanalyse und ist in Abbildung 6.1 dargestellt.

Somit werden im ersten Abschnitt die Systemkomponenten des Block-orientierten Modells vorgestellt. Insgesamt werden drei Modelle miteinander verglichen:

- Das Block-orientierte Modell der Ausgangssituation (Baseline)
- Die funktionale Erweiterung der Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer Autorität (Modell_1)
- Die Überarbeitung der funktionalen Erweiterung zur dynamischen Speicherallokation der Softwarekomponenten (Modell_2)

Der zweite Abschnitt beschreibt Entitäten der aktiven Akteure im Lebenszyklus des Modells. Diese werden nach Naturellen oder Maschinen unterteilt. Nachfolgend werden im dritten Abschnitt sicherheitsrelevante Anwendungsfälle (Use Cases) anhand des Produktlebenszyklus identifiziert. Dazu zählen die vier Phasen des Produktlebenszyklus, der Entwicklungs- und Fertigungsprozess, die Verwendung im Feld sowie Wiederaufbereitungsprozesse. Ziel ist es, das Verständnis des Systems durch seine Anwendungsfälle zu vertiefen, um die Eigenschaften des Block-orientierten Modells in den nachfolgenden Analysen zu fokussieren.

Die Zielsetzungen im vierten Abschnitt fassen die allgemeinen Sicherheitsziele und die im ersten Abschnitt sicherheitsrelevanten Systemkomponenten zusammen. Die Ziele ermöglichen in den folgenden Kapiteln die Identifikation von Sicherheitsbedrohungen und -anforderungen.

Der fünfte Abschnitt des Kapitels beschreibt die Bedrohungsanalyse. Hier sind Bedrohungsagenten definiert, um mögliche Angriffsvektoren zu identifizieren. Die Angriffsvektoren werden für die gesetzten Sicherheitsziele anhand eines Angriffsbaums bewertet. Zusätzlich werden die Bedrohungen in Relation zu den Systemkomponenten aus dem ersten Abschnitt gebildet.

Final wird im sechsten Abschnitt die Risikoanalyse dargestellt. Nach ISO 21434, ISO 26262, ISO 18045 und SAE J3061 werden Angriffs- und Schadenspotentiale in eine Risikobewertung integriert. Speziell werden die Potentiale anhand der in den vorherigen Abschnitten angeführten Sicherheitsziele, -analysen und -komponenten miteinander in Relation gesetzt. Aus diesen Konstellationen wird ein Schadenspotential berechnet.

Zusammengefasst werden im siebten Abschnitt die drei Block-orientierten Modelle miteinander verglichen und evaluiert. Hierbei werden diese Modelle hinsichtlich der Datenintegrität, der Zugriffskontrolle, kryptografischer Methoden und der sicheren Laufzeitumgebung bewertet. Der achte Abschnitt leitet die daraus resultierenden neuen Sicherheitsanforderungen ab.

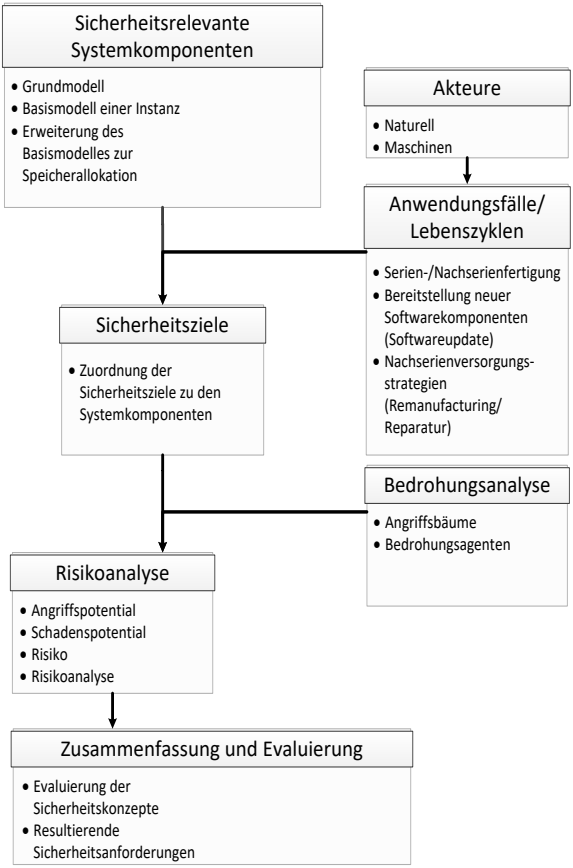


**Abbildung 6.1:** Sicherheitskonzept

## 6.1 Architekturübersicht und sicherheitsrelevante Systemkomponenten

Dieser Abschnitt beschreibt die Architekturübersicht und deren sicherheitsrelevante Systemkomponenten aus dem Leitbeispiel der drei Block-orientierten Modelle, die in Abbildung 6.2 beschrieben sind. Eine vollumfängliche T&R-Analyse betrachtet die sicherheitsrelevanten Systemkomponenten in verschiedenen Domänen. Diese unterteilen sich nach Modell in die unterschiedlichen Blocktypen (Bsp. *ROOT*) einer Instanz mit den zugehörigen, sicherheitsrelevanten Datenobjekten der Indizes (Bsp. *INDEX_STRUKTUR*) oder öffentlichen Schlüssel (Bsp. *INDEX_SCHLÜSSEL*). Außerdem wird die Eigenschaft angehängt, ob die Komponente signiert ($SIG$) ist somit einer Integritätsprüfung unterliegt. Eine im Root-Block befindliche, signierte Applikation des dritten Modelles wird somit bezeichnet als: *Modell_2.ROOT.APP.SIG*.

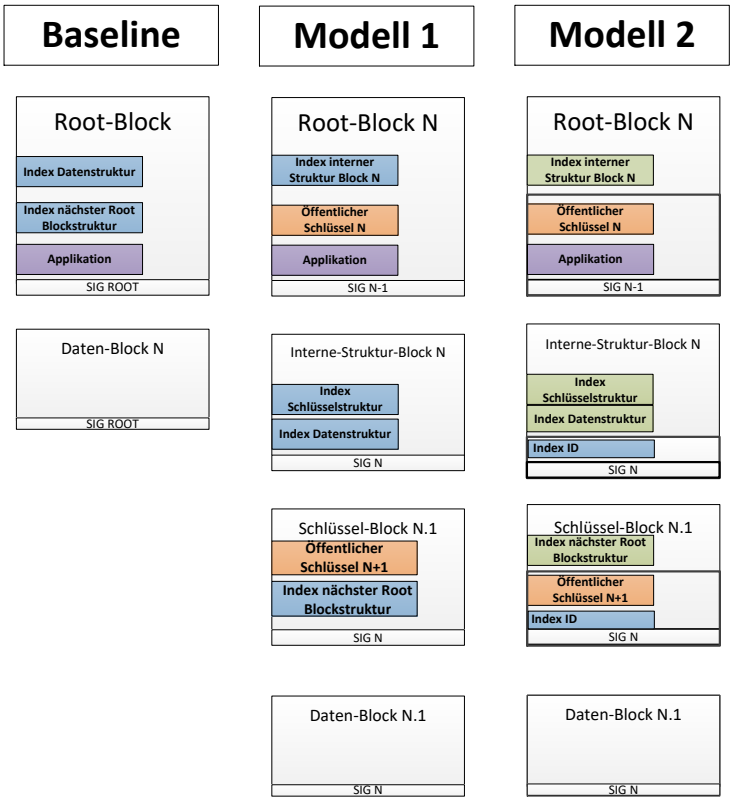


**Abbildung 6.2:** Sicherheitsrelevante Systemkomponenten

### 6.1.1 Block-orientiertes Modell der Ausgangssituation (Baseline)

Sämtliche Blöcke sind mit einem Root-Schlüssel signiert.

#### 6.1.1.1 Root-Block

*Baseline.ROOT.INDEX_DATEN.SIG:* Der Index für die interne Datenstruktur verweist auf die nachstehenden Datenblöcke, die der Root-Block verwendet. Zusätzlich sind hier auch IDs zur Softwareversionierung- und Referenzüberprüfung vorgesehen.

*Baseline.ROOT.INDEX_ROOT.SIG:* Der Index für die nächste Root-Block-Struktur verweist auf den Root-Block einer neuen Instanz. Zusätzlich sind hier auch IDs zur Softwareversionierung und Referenzüberprüfung vorgesehen. Dieser Index ist Bestandteil des signierten Root-Block-Bereiches.

*Baseline.ROOT.APP.SIG:* Die Applikation ist der lesbare Maschinencode und Bestandteil des signierten Root-Block-Bereiches.

#### 6.1.1.2 Daten-Block

*Baseline.DATEN.SIG:* Der Daten-Block ist immer mit dem Root-Schlüssel signiert und beinhaltet statische Datensätze.

### 6.1.2 Funktionale Erweiterung der Anforderung für die Updatefähigkeit einer Zielapplikation mit neuer Autorität (Modell_1)

#### 6.1.2.1 Root-Block

*Modell_1.ROOT.INDEX_STRUKTUR.SIG:* Dieser Index verweist auf den Interne-Struktur-Block und ist Bestandteil des signierten Root-Block-Bereiches.

*Modell_1.ROOT.INSTANZ_SCHLÜSSEL.SIG:* Der öffentliche Schlüssel wird zur Authentifizierung der in der Instanz befindlichen Blöcke verwendet und ist Bestandteil des signierten Root-Block-Bereiches.

*Modell_1.ROOT.APP.SIG:* Die Applikation ist der lesbare Maschinencode und ist Bestandteil des signierten Root-Block-Bereiches.

#### 6.1.2.2 Interne-Struktur-Block

*Modell_1.STRUKTUR.INDEX_SCHLÜSSEL.SIG:* Dieser Index stellt Informationen über die in der Instanz befindliche Struktur der Schlüssel-Blöcke bereit und ist Bestandteil des signierten Interne-Struktur-Block Bereiches.

*Modell_1.STRUKTUR.INDEX_DATEN.SIG:* Der Index für die interne Datenstruktur verweist auf die nachstehenden Datenblöcke, die der Root-Block verwendet. Zusätzlich sind hier auch IDs zur Softwareversionierung- und Referenzüberprüfung vorgesehen.

#### 6.1.2.3 Schlüssel-Block

*Modell_1.SCHLÜSSEL.BLOCK_SCHLÜSSEL.SIG:* Der eigenständige öffentliche Schlüssel wird zur Authentifizierung eines nächststehenden Root- oder Daten-Blockes verwendet und ist Bestandteil des signierten Schlüssel-Block Bereiches.

*Modell_1.SCHLÜSSEL.INDEX_BLOCK.SIG:* Dieser Index stellt Informationen für etwaige nächste Root- oder eigenständig signierte Daten-Blöcke bereit. Zusätzlich sind hier auch IDs zur Softwareversionierungs- und Referenzüberprüfung vorgesehen. Dieser Index ist ebenfalls Bestandteil des signierten Schlüssel-Block-Bereiches.

#### 6.1.2.4 Daten-Block

*Modell_1.DATEN.SIG:* Der Datenblock ist mit einem Instanz- oder einem eigenständigen Schlüsselpaar signiert und beinhaltet statische Datensätze.

### 6.1.3 Erweiterung zur dynamischen Speicherallokation der Softwarekomponenten (Modell_2)

#### 6.1.3.1 Root-Block

*Modell_2.ROOT.INDEX_STRUKTUR:* Dieser Index verweist auf den Interne-Struktur-Block und ist Bestandteil des Root-Blocks, jedoch außerhalb des signierten Bereiches.

*Modell_2.ROOT.INSTANZ_SCHLÜSSEL.SIG:* Der öffentliche Schlüssel wird zur Authentifizierung der in der Instanz befindlichen Blöcke verwendet, und ist Bestandteil des signierten Root-Block-Bereiches.

*Modell_2.ROOT.APP.SIG:* Die Applikation ist der lesbare Maschinencode und gehört zum signierten Root-Block-Bereich.

#### 6.1.3.2 Interne-Struktur-Block

*Modell_2.STRUKTUR.INDEX_SCHLÜSSEL:* Dieser Index stellt Informationen über die in der Instanz befindliche Struktur der Schlüsselblöcke bereit und ist dem Interne-Struktur-Block zugehörig, allerdings außerhalb des signierten Bereiches.

*Modell_2.STRUKTUR.INDEX_DATEN:* Der Index für die interne Datenstruktur verweist auf die nachstehenden Datenblöcke, die der Root-Block verwendet und ist Bestandteil des Interne-Struktur-Blocks, aber außerhalb des signierten Bereiches.

*Modell_2.STRUKTUR.INDEX_ID.SIG:* Der Index ID liefert zusätzlich IDs zur Software-versionierungs- und Referenzüberprüfung. Dieser Index ist Bestandteil des signierten Interne-Struktur-Block-Bereiches.

#### 6.1.3.3 Schlüssel-Block

*Modell_2.SCHLÜSSEL.INDEX_BLOCK:* Dieser Index stellt Informationen für etwaige nächste Root-, oder eigenständig signierte Daten-Blöcke

bereit und ist Bestandteil des Schlüssel-Block Bereiches, jedoch außerhalb des signierten Bereiches.

*Modell_2.SCHLÜSSEL.BLOCK_SCHLÜSSEL.SIG:* Der eigenständige öffentliche Schlüssel wird zur Authentifizierung eines nächststehenden Root- oder Daten-Blockes verwendet und ist Bestandteil des signierten Schlüssel-Block-Bereiches.

*Modell_2.STRUKTUR.INDEX_ID.SIG:* Der Index ID liefert zusätzlich IDs zur Software-versionierungs- und Referenzüberprüfung. Dieser Index ist Bestandteil des signierten Schlüssel-Block-Bereiches.

#### 6.1.3.4 Daten-Block

*Modell_2.DATEN.SIG:* Der Daten-Block ist mit einem Instanz- oder einem eigenständigen Schlüsselpaar signiert und beinhaltet statische Datensätze.

### 6.1.4 Sicherheitsrelevante Datenobjekte

Kapitel 4 beschreibt, dass die Architektur des i.MX6 den Hash des öffentlichen Schlüssels der Gerätedatei irreversibel in einem dafür vorgesehenen One-Time-Programmable-Register ablegt. Diese Funktionalität variiert in der Ausstattung der jeweils eingesetzten Hardwarearchitektur. Im Folgenden wird der öffentliche Schlüssel als Root-Schlüssel betrachtet, der irreversibel im Steuergerät abgelegt ist.

## 6.2 Akteure

Nachdem die Komponenten der verschiedenen Block-orientierten Modelle definiert wurden, werden im Folgenden die Akteure behandelt, die unter den beschriebenen Use Cases agieren. Hier wird zwischen Naturellen und Maschinen unterschieden.

### 6.2.1 Naturelle

- Techniker (Werk): Der Techniker im Werk bedient die Fertigungstester. Er verfügt nicht zwingend über ein tiefes Produkt-Knowhow hinsichtlich der Zielapplikation, ist jedoch ein Experte in der Bedienung und Konfigurierung des Fertigungstesters.
- Autorität für die Signierung der Softwarekomponenten: Hier handelt es sich um den Akteur, der die jeweiligen Softwareblöcke signiert und die Autorität (öffentlicher Schlüssel) bereitstellt. Die Autorität hat keinen Zugriff auf Fertigungseinrichtungen, dennoch die Berechtigung, Datensätze dem Trust Center bereitzustellen.
- Techniker (Servicewerkstatt): Dieser Akteur verwendet eine Diagnoseschnittstelle, um mit einem Diagnosetester ein Softwareupdate in der Werkstatt durchzuführen. Diese Software ist bereits im Vorfeld signiert.
- Endanwender: Der Endanwender hat nur Zugriff auf die Peripherie der Head Unit und Softwarekomponenten, die von der Zielapplikation bereitgestellt werden.

### 6.2.2 Maschinen

- Trust Center: Ein Trust Center (TC) liegt im gesicherten Bereich eines Produktherstellers oder Drittanbieters. Ein TC hat die Funktion, sämtliche Schlüsselmanagementaufgaben zu verrichten, was die Generierung von individuellen Signaturquellen und deren Signierungsverfahren für die jeweiligen Datenblöcke impliziert.
- Servicecenter: Ein Servicecenter kümmert sich um den Flottenbetrieb, indem es im Rahmen eines Service-orientierten Softwareupdates neue Versionen bereitstellt.
- Fertigungstester: Der Fertigungstester ist direkt mit einem Fertigungsnetzwerk verbunden. Er überträgt die Datensätze und Dateien des Trust Centers sowie Share Folder.
- Diagnosetester: Ein Diagnosetester ist mit einem Servicecenter der OEM verbunden. Er verfügt über erweiterte Berechtigungen, um Softwareupdates und Konfigurationen am Steuergerät durchzuführen. Unabhängige Diagnosetester, die nicht einem OEM zugeordnet sind, haben im Rahmen der vereinheitlichten Diagnosedienste nach ISO 14229 einen eingeschränkten Zugriff. Für Softwareupdates wird eine Pass-Thru-Möglichkeit etabliert, die eine Anbindung an ein OEM-Servicecenter erlaubt.
- Share Folder: Ein Share Folder ist ein im Fertigungsnetzwerk oder Servicecenter gesicherter Massenspeicher, der sämtliche signierten Programm-/Datenblöcke respektive Softwarecontainer verwaltet.

- Datenbanken: Datenbankstrukturen kommen sowohl im Trust Center und Servicecenter wie auch als virtuell angebundene Instanzen des Fertigungstesters zum Einsatz. Auf ihnen sind gesicherte/verschlüsselte Messergebnisse und Autoritäten hinterlegt.

## 6.3 Sicherheitsrelevante Anwendungsfälle

Im folgenden Abschnitt werden Anwendungsfälle in Form von Use Cases beschrieben. Der Fokus liegt auf den Sicherheitsaspekten zum Handling der signierten Softwareblöcke. Im Kern werden anhand des Produktlebenszyklus drei Abschnitte definiert, die sich aus der Entwicklung und Fertigung, der Verwendung im Feld und der Wiederaufbereitung/Reparatur zusammensetzen.

Neben den im zweiten Abschnitt definierten Akteuren werden Konditionen und Voraussetzungen des Prozesses dargestellt. Dies umfasst die Prozessbeschreibung mit Fehleranwendungsfällen. Hierbei werden mögliche Bedrohungen im Rahmen des Prozesses bereits kurz dargestellt.

### 6.3.1 Entwicklungs- und Fertigungsprozess

Der Fertigungsprozess impliziert die Softwareentwicklung mit Signierung der Softwarekomponenten nach Tabelle 6.1 und der Fertigung des Steuergerätes nach Tabelle 6.2. Folglich kommt es im Fertigungsprozess zur Installation der signierten Softwarekomponenten in die Speichereinheiten des zu fertigenden Steuergerätes.

**Tabelle 6.1:** Anwendungsfall: Entwicklung und Signierung

| **UC.Entwicklung.Signierung** | |
|---|---|
| Beschreibung | Dieser Use Case beschreibt den Signierungsprozess einer Autorität. |
| Akteure | • Autorität für die Signierung der Softwarekomponenten<br>• Trust Center<br>• Datenbanken<br>• Share Folder |
| Enthaltene Prozesse | Keine |
| Voraussetzung | - |
| Normaler Prozess | Im Signierungsprozess wird eine Instanz mit den jeweiligen Blöcken signiert. Hierzu wird der im Trust Center und ggf. in den Datenbankstrukturen abgelegte private Schlüssel verwendet, um den jeweiligen Block zu signieren. Der öffentliche Schlüssel wird dann mit der signierten Instanz/ den Blöcken den Bedarfsträgern übergeben und in einem Share Folder abgelegt. |
| Alternativ bei Fehler | Bei auftretendem Fehler wird dieser Prozess wiederholt. |
| Mögliche Bedrohungen | • Manipulierung der Datenbanken<br>• Manipulierung des Trust Centers<br>• Manipulierung des Share Folder |
| Sicherheitsziele | Software-Integrationsschutz |

**Tabelle 6.2:** Anwendungsfall: Fertigung

| **UC.Fertigung** | |
|---|---|
| Beschreibung | Dieser Use Case beschreibt den Fertigungsprozess des Gerätes. |
| Akteure | • Techniker (Werk)<br>• Fertigungstester<br>• Share Folder |
| Enthaltene Prozesse | Keine |
| Voraussetzung | UC.Entwicklung.Signierung |
| Normaler Prozess | Im Fertigungsprozess kommt es zur ersten Installation der ersten Instanz in den Speichereinheiten des zu fertigenden Gerätes. Diese Instanz wurde mit dem Root Key signiert. Man spricht hier von einem Pre-Programming, der In-Circuit vorgenommen wird. Hierbei werden die im PCB befindlichen Controller über eine Nadelkontaktierung angeschlossen und restliche Konfigurationen vorgenommen sowie der öffentliche Schlüssel des Root Key gesetzt.<br>Neben einer kompletten elektrischen Prüfung wird das Gerät im End-of-Line-Test finalisiert. Dies bedeutet, dass nun auch die folgenden Instanzen installiert werden und das Gerät verschlossen wird. Das Verschließen findet statt, indem der öffentliche Schlüssel des Root Key irreversibel gesetzt wird.<br>Es folgt die Lieferung in ein Fertigungswerk zum OEM oder in ein Zwischenlager zum Vertrieb für Servicewerkstätten. |
| Alternativ bei Fehler | Bei auftretendem Fehler wird das Gerät verschrottet. |
| Mögliche Bedrohungen | • Manipulierung des Share Folder<br>• Manipulierung des Datensatzes für den öffentlichen Schlüssel des Root Keys während der Datenübertragung im Fertigungsprozess<br>• Manipulation eines zu fertigen Gerätes vor EOL |
| Sicherheitsziele | Aktivierung des Secure-Boot-Zustandes |

### 6.3.2 Verwendung im Feld

Für funktionale Erweiterungen und insbesondere auch für sicherheitsbezogene Patches ist die Implementierung neuer Softwarekomponenten für die Nachserie ein wichtiger Bestandteil moderner Servicestrategien. Um Softwarepakete auch von Drittparteien einbinden zu können, sind im Folgenden in Tabelle 6.3 und Tabelle 6.4 zwei Use Cases definiert, die ein Softwareupdate over the Air (FOTA) und in einer Servicewerkstatt beschreiben.

**Tabelle 6.3:** Anwendungsfall: Softwareupdate Flash over the Air

| **UC.Softwareupdate.FOTA** | |
|---|---|
| Beschreibung | SW-Update eines Softwareblockes / Instanz im Feld |
| Akteure | • Share Folder<br>• Servicecenter<br>• Endanwender |
| Enthaltene Prozesse | Keine |
| Voraussetzung | UC.Fertigung |
| Normaler Prozess | Der OEM oder ein Flottenbetreiber stellt das jeweilige Release für einen Download außerhalb des Werkes zur Verfügung. Ein Software Management Center verwaltet die angestoßenen Updates. Softwareupdates wie Kartenmaterial oder Applikations-Updates werden in Installationspaketen vorbereitet und können beispielsweise einen neuen signierten Datenblock oder eine neue Instanz darstellen. Definierte Softwarepakete werden in Abhängigkeit zu der bereits verwendeten Software erstellt. Das Update kann physikalisch über ein Speichermedium oder drahtlos over The Air geschehen.<br>*Flash over The Air*: Bei einem Softwareupdate over The Air wird das zu installierende SW-Paket über viele kleine inkrementelle Softwarepakete zum Steuergerät übertragen und im zugehörigen Datenspeicher zwischengespeichert. Erst wenn die Manifest-Datei komplett übertragen wurde, kann der Installationsprozess in der Applikation gestartet werden.<br>*Softwareupdate über ein physikalisches Medium*: Ein Softwareupdate über ein physikalisches Medium, wie z. B. einen USB-Stick, wird durch eine Anfrage, z. B. eines Serviceportals, angeregt. Das Software Management Center generiert das Manifest und stellt es über ein Kundenportal zur Verfügung. |
| Alternativ bei Fehler | SW-Updateprozess wird abgebrochen. |
| Mögliche Bedrohungen | Bypass, Seitenkanalangriff |
| Sicherheitsziele | Sicherstellung eines Softwareupdates mit signiertem Softwareblock mit Integritätsprüfung |

**Tabelle 6.4:** Anwendungsfall: Softwareupdate in einer Servicewerkstatt

| **UC.Softwareupdate.Servicewerkstatt** | |
|---|---|
| Beschreibung | SW-Update eines Softwareblockes/Instanz in einer Servicewerkstatt |
| Akteure | • Share Folder<br>• Servicecenter<br>• Diagnosetester<br>• Techniker (Servicewerkstatt) |
| Enthaltene Prozesse | Keine |
| Voraussetzung | UC.Fertigung |
| Normaler Prozess | Der OEM, Flottenbetreiber oder die Drittpartei stellt das jeweilige Release für einen Download außerhalb des Werkes zur Verfügung. Ein Software Management Center verwaltet die angestoßenen Updates. Ein Servicetechniker stößt nun über einen Diagnosetester ein Softwareupdate mittels einer physikalischen Diagnoseschnittstelle an. |
| Alternativ bei Fehler | SW-Updateprozess wird abgebrochen. |
| Mögliche Bedrohungen | • Bypass, Seitenkanalangriff<br>• Missbrauch der Berechtigungskonzepte des Diagnosetesters durch den Servicetechniker |
| Sicherheitsziele | Sicherstellung eines Softwareupdates mit signiertem Softwarecontainer mit Integritäts- und elektrischer Prüfung |

### 6.3.3 Wiederaufbereitung und Reparatur

Ein Remanufacturing-Prozess ist ein Wiederaufbereitungsprozess, in dem das Produkt als Schadteil aus dem Feld kommt und es qualitativ nach serienproduktionstechnischen Standards wiederaufbereitet und werkseitig rückgesetzt wird. Somit wird zu einer zeitwertgerechten, ressourceneffizienten Kreislaufwirtschaft beigetragen, da die reparierten und wiederaufbereiteten Steuergeräte erneut in den gleichen oder anderen Systemen Verwendung finden können.

Den Counterpart dazu bildet die Individualreparatur, wobei das reparierte Steuergerät im jeweils alten Zielsystem weiterhin angewendet wird und es zu keiner Änderung der Softwarestände kommt. Einen Sonderfall stellt hierbei eine sogenannte „Flash vor Versand“(FvV)-Variante dar, in der alte Datensätze in ein neues Steuergerät portiert werden.

Im Folgenden werden zwei Use Cases in Tabelle 6.5 und Tabelle 6.6 vorgestellt, der Wiederaufbereitungsprozess und die FvV-Variante.

**Tabelle 6.5:** Anwendungsfall: Wiederaufbereitungsprozess

| **UC.REMAN** | |
|---|---|
| Beschreibung | Komplettes werkseitiges Rücksetzen |
| Akteure | • Techniker (Werk)<br>• Fertigungstester<br>• Share Folder |
| Enthaltene Prozesse | Teilbereiche der UC.Fertigung |
| Voraussetzung | • UC.Entwicklung.Signierung<br>• UC.Fertigung |
| Normaler Prozess | Werkseitige Rücksetzung des Steuergerätes. Hierbei wird ein Fertigungsserver verwendet, der Teilbereiche des Fertigungsprozesses wiederholt, um den gewünschten Softwarestand zu erreichen. |
| Alternativ bei Fehler | Wiederholung, ggf. Verschrottung |
| Mögliche Bedrohungen | Manipulierung des Fertigungsshares |
| Sicherheitsziele | Sicherstellung eines Softwareupdates mit signiertem Softwarecontainer mit Integritäts- und elektrischer Prüfung |

**Tabelle 6.6:** Anwendungsfall: Flash vor Versand

| **UC.FvV** | |
|---|---|
| Beschreibung | Portierung eines Softwarestandes zu einem Neugerät |
| Akteure | • Techniker (Werk)<br>• Fertigungstester<br>• Servicecenter<br>• Share Folder |
| Enthaltene Prozesse | • Teilbereiche der UC.Fertigung<br>• UC.Softwareupdate.Servicewerkstatt |
| Voraussetzung | • UC.Entwicklung.Signierung<br>• UC.Fertigung |
| Normaler Prozess | Dieser Sonderfall der Individualreparatur erfordert die Voraussetzung, dass über einen sogenannten Geräterohling ein spezifischer Softwarestand erreicht wird, der nicht der werkseitigen Versionierung entspricht. Über ein Datenabbild des alten Gerätes werden die jeweiligen Softwareblöcke in den Speicher des neuen Gerätes portiert. Dazu ist es notwendig, eine Verheiratung der Root-Instanz und der Folgeinstanzen auszuführen. |
| Alternativ bei Fehler | Wiederholung, ggf. Verschrottung |
| Mögliche Bedrohungen | • Manipulierung des Fertigungsshares<br>• Bypass, Seitenkanalangriff |
| Sicherheitsziele | Software-Integritätsprüfung |

## 6.4 Sicherheitsziele und zugehörige Objekte

Im folgenden Abschnitt werden Sicherheitsziele für die Risikoanalyse definiert. Aus den sicherheitsrelevanten Use Cases werden die möglichen Bedrohungen zu den sicherheitsrelevanten Systemkomponenten abgeleitet.

### 6.4.1 Zugriffskontrolle

Authentifizierungs- und Autorisierungsmechanismen für relevante Schnittstellen/Dienste oder deren Freischaltungen müssen im System gegeben sein und implizieren die Rechteverwaltung eines Root-Blockes. Üblicherweise besteht ein Autorisierungsmechanismus aus einer SEED&KEY-Vorschrift oder über die Auswertung einer signierten Botschaft mittels eines asymmetrischen Schlüsselverfahrens.

### 6.4.2 Datenintegrität

Für den Schutz der Integrität der Software müssen analog zu den CIA Triads nach [41] und [1] Maßnahmen vorgesehen werden, um eine unbefugte Änderung zu vermeiden.

### 6.4.3 Kryptografische Methoden

Es müssen angemessene kryptografische Algorithmen verwendet werden, um die Infrastruktur der öffentlichen Schlüssel, aber auch Datensätze

schützen zu können.

### 6.4.4 Sichere Laufzeitumgebung

Die jeweilige Instanz des Root-Blockes darf weder manipulierte Softwareelemente ausführen, noch die Ausweitung von undefinierten Speicherplätzen zulassen.

### 6.4.5 Zuordnung von Zielen und Komponenten

Die im vorherigen Abschnitt festgelegten sicherheitsrelevanten Systemkomponenten geben allgemeine Ziele und Ambitionen für die Updatefähigkeit einer Zielapplikation mit neuer Autorität wieder. Dieser Abschnitt ordnet die Sicherheitsobjekte den Sicherheitskomponenten zu. Jede dieser Beziehungen ist in Tabelle 6.7 mit einem „X“ gekennzeichnet.

Die Zugriffskontrolle lässt sich auf alle Applikationen anwenden, die über einen Autorisierungsmechanismus verfügen, um beispielsweise Diagnosefunktionen oder Softwareupdates freizuschalten.

Die Datenintegrität findet auf alle signierten Komponenten Anwendung. Hierzu zählt auch der Root-Schlüssel des Systems. Ferner sind auch die nicht signierten Indizes des dritten Modells betroffen, da sie einerseits auf signierte Komponenten verweisen, andererseits ausschließlich in der Laufzeit von signierten Applikationen verändert werden sollen. Auch spielen Vererbungsmechanismen zur Freischaltung von Adressbereichen eine wesentliche Rolle.

**Tabelle 6.7:** Zuordnung der Sicherheitsobjekte zu den Sicherheitskomponenten

| Sicherheitsobjekte | Sicherheitsrelevante Systemkomponenten | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Baseline | | | | Modell_1 | | | | | | | | | Modell_2 | | | | | | | | | | |
| | Baseline.ROOT.INDEX_DATEN.SIG | Baseline.ROOT.INDEX_ROOT.SIG | Baseline.ROOT.APP.SIG | Baseline.DATEN.SIG | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.SIG | Modell_1.ROOT.INDEX_STRUKTUR.SIG | Modell_1.ROOT.APP.SIG | Modell_1.STRUKTUR. INDEX_SCHLÜSSEL.SIG | Modell_1.STRUKTUR. INDEX_DATEN.SIG | Modell_1.SCHLÜSSEL.BLOCK-SCHLÜSSEL.SIG | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SIG | Modell_1.DATEN.SIG | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.SIG | Modell_2.ROOT.INDEX_STRUKTUR | Modell_2.ROOT.APP.SIG | Modell_2.STRUKTUR.INDEX_SCHLÜSSEL | Modell_2.STRUKTUR.INDEX_DATEN | Modell_2.STRUKTUR.INDEX_ID.SIG | Modell_2.SCHLÜSSEL.BLOCK-SCHLÜSSEL.SIG | Modell_2.SCHLÜSSEL.INDEX_BLOCK | Modell_2.SCHLÜSSEL. INDEX_ID.SIG | Modell_2.DATEN.SIG |
| Zugriffskontrolle | | | x | x | x | | x | | | x | | x | x | | x | | | | x | | | | x |
| Datenintegrität | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| Kryptografische Methoden | | | x | x | | | x | | | | | x | | | x | | | | | | | | x |
| Sichere Laufzeitumgebung | x | x | x | x | | x | x | x | x | | x | x | | x | x | x | x | x | | x | x | x | x |

Kryptografische Methoden finden in den Applikationen Anwendung. Auch sind Datensätze betroffen, da hier unter Umständen zugehörige Berechnungsvorschriften oder statische Bibliotheken untergebracht sind und in die Applikation eingebunden werden können.

Eine sichere Laufzeitumgebung betrifft neben der Applikation auch die zugehörigen Indizes für die Bereitstellung eines Adressbereiches. Auch die Root-Instanz ist von der sicheren Laufzeitumgebung berührt, da diverse Architektur-bezogene Register die Laufzeitumgebung beeinflussen.

## 6.5 Bedrohungsanalyse

Im ersten Abschnitt wurden die sicherheitsrelevanten Systemkomponenten definiert. Dieser Abschnitt zeigt nun konkrete Angriffsmöglichkeiten zu den Komponenten, indem ein sogenannter Angriffsbaum beschrieben ist. Jeder Pfad in einem solchen Baum entspricht einem bestimmten Angriff. Um im weiteren Schritt die Ermittlung der Angriffs- und Schadenspotentiale zu erleichtern, ist in jedem Angriffsbaum eine Tabelle aller zugehörigen Komponenten für jeden Angriffspfad aufgelistet. Vorab werden analog zu den beschriebenen Akteure nun Bedrohungsagenten definiert. Diese werden ebenfalls für die Risikoanalyse benötigt.

### 6.5.1 Angriffsbäume

Dieser Abschnitt enthält die Auflistung der Angriffsbäume für die Sicherheitsobjekte. Diese bilden die Grundlage der Risikoanalyse. Die Angriffsszenarien wurden aus den Anwendungsfällen und Zielsetzungen sowie vom allgemeinen Kontext extrahiert. Bekannte Angriffsvektoren und Schwachstellen aus gegebenen Architekturkomponenten wurden ebenfalls in die Betrachtung einbezogen. Somit können alle betroffenen Sicherheitskomponenten bei der Durchführung eines Angriffs dargestellt werden.

#### 6.5.1.1 Zugriffskontrolle

Abbildung 6.3 zeigt den Angriffsbaum für das Sicherheitsobjekt der Zugriffskontrolle. Im Kern sind fünf mögliche Angriffsszenarien definiert.

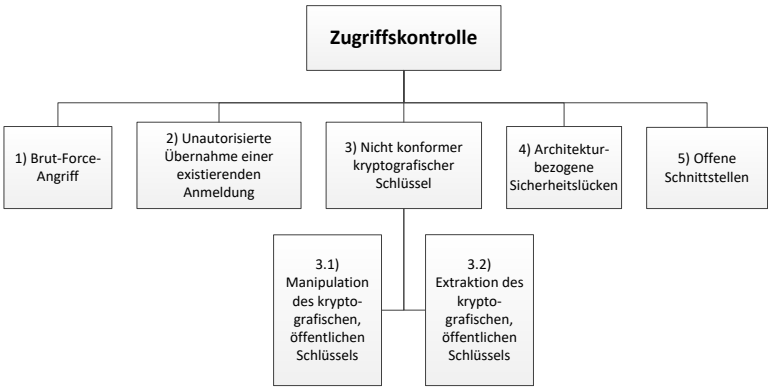


**Abbildung 6.3:** Angriffsbaum „Zugriffskontrolle“

1) Brute-Force-Angriff: Der Angreifer versucht sequenziell alle möglichen Kombinationen, die ein Login erfordern. Beispielsweise kann dies in Form eines Passwortes oder der Berechnungsvorschrift einer Zufallszahl geschehen. Bei einem asymmetrischen Berechtigungskonzept stellt dies die korrekte Signatur eines Prüftelegramms dar. Der Angriff wird so lange wiederholt, bis eine Zufallskollision des richtigen Datums erreicht ist.

2) Unautorisierte Übernahme einer existierenden Anmeldung: Hier könnte sich beispielsweise eine dritte Partei über einen Man-in-the-Middle-Angriff unerlaubterweise Zugang zu einer bereits existierenden Anmeldesession verschaffen. Wenn z. B. der Authentifizierungsvorgang eines Softwareupdates vollzogen ist, könnte gezielt Schadsoftware anstelle der Zielapplikation übertragen werden.

3) Nicht konformer kryptografischer Schlüssel:

3.1) Manipulation des kryptografischen Schlüssels: Im Rahmen einer Konsistenzprüfung könnte der gesicherte, öffentliche Schlüssel so manipuliert werden, dass manipulierte Software mit einer spezifischen Signatur für gültig empfunden werden kann. Dies schließt auch das gezielte Löschen oder Ersetzen einer Signaturquelle mit ein.

3.2) Extraktion des kryptografischen Schlüssels: Wenn ein kryptografischer, öffentlicher Schlüssel extrahiert wird, so kann mit keiner Kollisionsprüfung aus 1) eine Signatur für eine Schadsoftware ermittelt werden.

4) Architekturbezogene Sicherheitslücken: Beispielsweise können Rechnerarchitekturbezogene Sicherheitslücken ausgenutzt werden, um Sonderfunktionen oder Manipulationen vornehmen zu können.

5) Offene Schnittstellen: Im Sicherheitskonzept nicht beachtete Schnittstellen, wie z. B. eine offene JTAG-Schnittstelle, können zusätzliche Funktionen bereitstellen, die so in der Zielapplikation nicht vorgesehen sind.

Im Folgenden werden nun in Tabelle 6.8 die Sicherheitsobjekte der Zugriffskontrolle den sicherheitsrelevanten Systemkomponenten zugeordnet.

**Tabelle 6.8:** Zuordnung der Sicherheitsobjekte „Zugriffskontrolle“ zu den sicherheitsrelevanten Systemkomponenten

| Angriffsbaum | | | Sicherheitsrelevante Systemkomponenten | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Baseline | | | Modell_1 | | | | Modell_2 | | | |
| | | | ROOT.SCHLÜSSEL | Baseline.ROOT.APP.SIG | Baseline.DATEN.SIG | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.SIG | Modell_1.ROOT.APP.SIG | Modell_1.SCHLÜSSEL.BLOCK-SCHLÜSSEL.SIG | Modell_1.DATEN.SIG | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.SIG | Modell_2.ROOT.APP.SIG | Modell_2.SCHLÜSSEL.BLOCK-SCHLÜSSEL.SIG | Modell_2.DATEN.SIG |
| Zugriffskontrolle | 1 | Brut-Force-Angriff | | x | | | x | | | | x | | |
| | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | | x | x | | x | | x | | x | | x |
| | 3.1 | Manipulation des kryptografischen Schlüssels | x | | | x | | x | | x | | x | |
| | 3.2 | Extraktion des kryptografischen Schlüssels | x | | | x | | x | | x | | x | |
| | 4 | Architekturbezogene Sicherheitslücken | x | x | | | x | | | | x | | |
| | 5 | Offene Schnittstellen | | x | | | x | | | | x | | |

#### 6.5.1.2 Datenintegrität

Abbildung 6.4 zeigt den Angriffsbaum für das Sicherheitsobjekt der Datenintegrität. Hier wurden drei Angriffsszenarien deklariert.

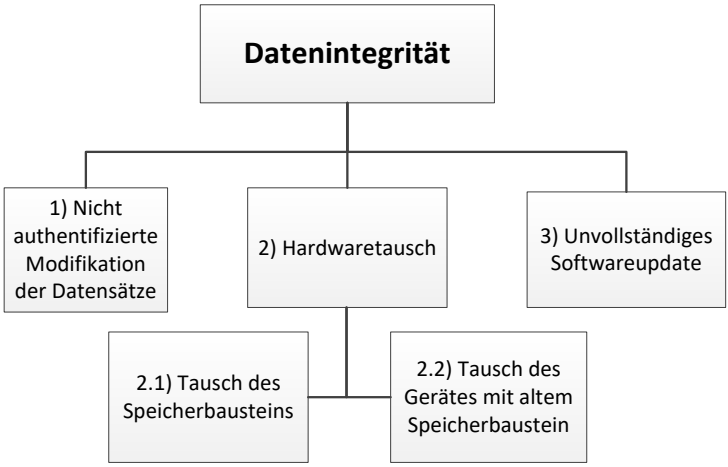


**Abbildung 6.4:** Angriffsbaum Datenintegrität

1) Nicht authentifizierte Modifikation der Datensätze: Das Szenario betrachtet einen Angreifer, der Softwaredatensätze im statischen Datenbereich ändert. So könnten abgelegte Zertifikate manipuliert oder es kann sich über Änderungen direkter Zugriff auf Hardwareschnittstellen beschafft werden. Hierbei sind alle in einem Speicherbaustein befindlichen Daten affektiert.

2) Hardwaretausch: Hier kommen zwei Szenarien in Betracht:

2.1) Tausch des Speicherbausteines: Auf Leiterplattenebene ersetzt ein Angreifer beispielsweise ein externes Flash-Speicher-Bauteil. Dieser neue Speicherbaustein könnte manipulierte Software beinhalten.

2.2) Tausch des Gerätes: Hier setzt ein Angreifer einen alten Speicherbaustein in ein neues Gerät ein. Die Software des alten Speicherbausteines könnte Funktionen beinhalten, die so nicht für den Einsatz des neuen Gerätes vorgesehen sind.

3) Unvollständiges SW-Update: Wenn ein Softwareupdateprozess schlecht ausgelegt ist und ein Angreifer somit Sicherheitsanfälligkeiten ausnutzt.

Im Folgenden werden nun in Tabelle 6.9 die Sicherheitsobjekte den sicherheitsrelevanten Systemkomponenten zugeordnet.

**Tabelle 6.9:** Zuordnung der Sicherheitsobjekte „Datenintegrität“ zu den sicherheitsrelevanten Systemkomponenten

| Angriffsbaum | | | Sicherheitsrelevanten Systemkomponenten | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Baseline | | | | | | Modell_1 | | | | | | | | | | Modell_2 | | | | | | | | | | | | |
| | | | ROOT.SCHLÜSSEL | Baseline.ROOT.INDEX_DATEN.SIG | Baseline.ROOT.INDEX_ROOT.SIG | Baseline.ROOT.APP.SIG | Baseline.DATEN .SIG | Modell_1.ROOT.INSTANZ.SCHLÜSSEL.SIG | Modell_1.ROOT.INDEX_STRUKTUR.SIG | Modell_1.ROOT.APP.SIG | Modell_1.STRUKTUR_INDEX_SCHLÜSSEL.SIG | Modell_1.STRUKTUR_INDEX_DATEN.SIG | Modell_1.SCHLÜSSEL.BLOCK.SCHLÜSSEL.SIG | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SIG | Modell_1.DATEN.SIG | Modell_2.ROOT.INSTANZ.SCHLÜSSEL.SIG | Modell_2.ROOT.INDEX_STRUKTUR | Modell_2.ROOT.APP.SIG | Modell_2.STRUKTUR.INDEX_.SCHLÜSSEL | Modell_2.STRUKTUR.INDEX_DATEN | Modell_2.STRUKTUR.INDEX_.ID.SIG | Modell_2.SCHLÜSSEL.BLOCK.SCHLÜSSEL.SIG | Modell_2.SCHLÜSSEL.INDEX_BLOCK | Modell_2.SCHLÜSSEL .INDEX_.ID.SIG | Modell_2.DATEN.SIG |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | 2.1 | Tausch des Speicherbausteines | | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | 2.2 | Tausch des Controllers | x | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 3 | Unvollständiges SW-Update | | | | x | x | | | x | | | | | | x | | | x | | | | | | | | | | x |

#### 6.5.1.3 Kryptografische Methoden

Abbildung 6.5 zeigt den Angriffsbaum für das Sicherheitsobjekt der kryptografischen Methoden. Hier wurden zwei Angriffsszenarien ausgemacht.

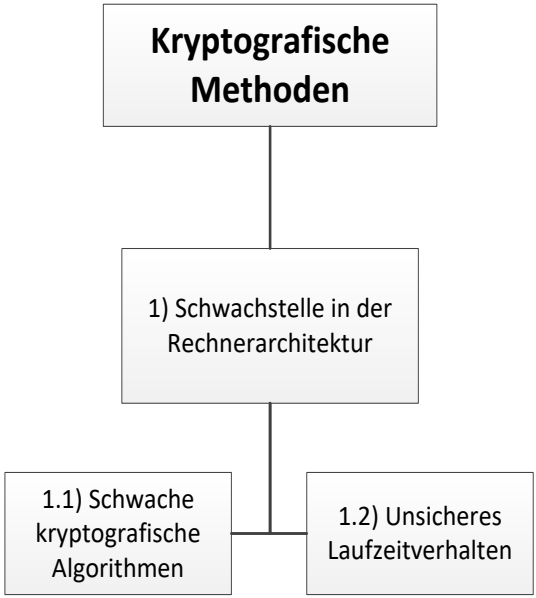


**Abbildung 6.5:** Angriffsbaum „Kryptografische Methoden“

1) Schwachstelle in der Rechnerarchitektur: Die interne Verarbeitung von kryptografischen Berechnungen kann durch einen Angreifer gestört oder ausgelesen werden, wenn das Sicherheitsdesign der Rechnerarchitektur

nicht ausreichend geschützt ist. Dies ist auf zwei Ursachen zurück zu führen.

1.1) Schwache kryptografische Algorithmen

1.2) Unsicheres Laufzeitverhalten

Im Folgenden werden nun in Tabelle 6.10 die Sicherheitsobjekte den sicherheitsrelevanten Systemkomponenten zugeordnet.

**Tabelle 6.10:** Zuordnung der Sicherheitsobjekte „Kryptografische Methoden“ zu den sicherheitsrelevanten Systemkomponenten

| Angriffsbaum | | | Sicherheitsrelevanten Systemkomponenten | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Baseline | | | Modell_1 | | | Modell_2 | | |
| | | | ROOT.SCHLÜSSEL | Baseline.ROOT.APP.SIG | Baseline.DATEN.SIG | Modell_1.ROOT.APP.SIG | Modell_1.DATEN.SIG | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.SIG | Modell_2.ROOT.APP.SIG | Modell_2.DATEN.SIG | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.SIG |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | x | x | x | x | x | x | x | x | x |
| | 1.2 | Unsicheres Laufzeitverhalten | x | x | | x | | | x | | |

#### 6.5.1.4 Sichere Laufzeitumgebung

Abbildung 6.6 zeigt den Angriffsbaum für das Sicherheitsobjekt der kryptografischen Methoden. Hier wurden vier Angriffsszenarien ausgemacht.

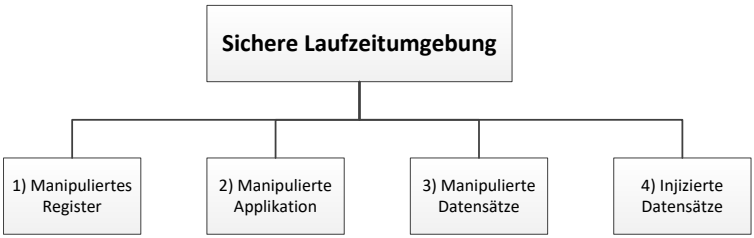


**Abbildung 6.6:** Angriffsbaum „Sichere Laufzeitumgebung“

1) Manipuliertes Register: Durch eine Manipulation in der Root-Ebene kann ein Angreifer von einem Standort booten.

2) Manipulierte Applikation: Der Angreifer führt manipulierte Applikationen aus. Dieser Angriff setzt jedoch voraus, dass der Angreifer die Integrität eines Root-Blockes erfolgreich manipuliert hat.

3) Manipulierte Datensätze: Der Angreifer führt manipulierte Datensätze aus. Dieser Angriff setzt jedoch voraus, dass der Angreifer die Integrität eines Daten-Blockes erfolgreich manipuliert hat.

4) Injizierte Datensätze: Der Angreifer kann die Sicherheitsanfälligkeit innerhalb einer Applikation ausnutzen, indem er für vorgesehene Indizes manipuliert.

Im Folgenden werden nun in Tabelle 6.11 die Sicherheitsobjekte den sicherheitsrelevanten Systemkomponenten zugeordnet.

**Tabelle 6.11:** Zuordnung der Sicherheitsobjekte „Sichere Laufzeitumgebung“ zu den sicherheitsrelevanten Systemkomponenten

| Angriffsbaum | | | Sicherheitsrelevanten Systemkomponenten | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Baseline | | | | | Modell_1 | | | | | | Modell_2 | | | | | | |
| | | | Baseline.ROOT.INDEX_DATEN.SIG | Baseline.ROOT.INDEX_ROOT.SIG | Baseline.ROOT.APP.SIG | Baseline.DATEN.SIG | Modell_1.ROOT.INDEX_STRUKTUR.SIG | Modell_1.ROOT.APP.SIG | Modell_1.STRUKTUR.INDEX_SCHLÜSSEL.SIG | Modell_1.STRUKTUR.INDEX_DATEN.SIG | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SIG | Modell_1.DATEN.SIG | Modell_2.ROOT.INDEX_STRUKTUR | Modell_2.ROOT.APP.SIG | Modell_2.STRUKTUR.INDEX_SCHLÜSSEL | Modell_2.STRUKTUR.INDEX_DATEN | Modell_2.STRUKTUR.INDEX_ID.SIG | Modell_2.SCHLÜSSEL.INDEX_BLOCK | Modell_2.DATEN.SIG |
| Sichere Laufzeitumgebung | 1 | Manipuliertes Register | | | | | | | | | | | | | | | | | | |
| | 2 | Manipulierte Applikation | | | x | | | x | | | | | | | x | | | | | |
| | 3 | Manipulierte Datensätze | | | | x | | | | | | x | | | | | | | | x |
| | 4 | Injizierte Datensätze | x | x | | | x | | x | x | x | | x | | x | x | x | x | x | |

## 6.5.2 Bedrohungsagenten

In diesem Abschnitt werden in Tabelle 6.12 mögliche Angriffsentitäten in Form von Bedrohungsagenten beschrieben sowie Kompetenzen und Zugangsmöglichkeiten zugeordnet. Diese Einteilung dient als Orientierungshilfe sowie Ausgangspunkt für die Identifikation von potentiellen Angriffen und zur Bewertung des Risikos.

**Tabelle 6.12:** Bedrohungsagenten

| ID | Information | Typische Expertise | Exemplarische Angriffsszenarien |
|---|---|---|---|
| **Extern** | Ein externer Angreifer hat keinen physikalischen Zugang zu dem Gerät. | Üblicherweise Zugriff auf WiFi/NFC-Schnittstellen | Datenmanipulation im Rahmen eines Man-in-the-middle-Angriffes, Kryptoanalysen |
| **Intern** | Angreifer hat physischen Zugang zum Gerät und zu sämtlichen Schnittstellen und Hardwarekomponenten. | Zum Teil detailliertes Insiderwissen und Gebrauch von Hacking/Sondertools. | Manipulation der Softwarestände und Schnittstellen, Auslesen von Speicherinhalten.<br>Man-in-the-middle-Angriffe bei Hardware-schnittstellen. |
| **Insider/Angestellter** | Angestellter mit internen und vertraulichen Informationen | Detaillierte Insiderinformationen, interne Tools und Zugangsberechtigungen zu Portalen | Siehe Intern. Hinzu kommen Zugangsberechtigungen zu Fertigungs-technischen Tools und Einrichtungen. |
| **Kunden-Portal** | In diesem Portal liegen die signierten SW-Stände der Applikationen. Mitarbeiter/Insider haben Zugang zu diesen SW-Ständen. | Fundiertes Wissen über das Produkt und Diagnosetester | Datenmanipulation.<br>Man-in-the-middle-Angriffe über physikalische Leitungen,<br>Kryptoanalysen. |
| **Fertigungsportal (Trust Center)** | In diesem Portal/dieser Datenbankstruktur liegen private Schlüsselsätze und Signierungsquellen sowie streng vertrauliche Informationen.<br>Mitarbeiter/Insider haben Zugang zu diesen Trust Center. | Fundiertes Wissen über die Fertigungs-einrichtungen. Gebrauch von internen Fertigungstools. | Kritische Daten- und Key-Manipulation |

## 6.6 Risikoanalyse

Dieses Kapitel bewertet final das Angriffs- und Schadenspotential, die mit der Bedrohungsanalyse aus Abschnitt 6.5 verbunden sind. Grundlage hierfür bilden die Angriffsbäume und deren resultierende Angriffsszenarien sowie die beschriebenen Bedrohungsagenten.

Der erste Abschnitt legt das Angriffspotential und dessen Quantifizierung fest. Hier werden grundlegende Definitionen nach ISO 18045 [115] erstellt. Dazu gehören zeitliche Angriffsaspekte, -equipment und Bedrohungsagenten.

Der zweite Abschnitt beschreibt das Schadenspotential, das unter Safety- und funktionellen Betrachtungen ermittelt wird.

Das resultierende Risiko ergibt sich aus der Kombination des Angriffs- und Schadenspotentials. Hierzu wird eine Grundlage im dritten Abschnitt erstellt. Final wird eine Zusammenfassung vorgenommen, in der eine Risikobewertung der erstellten Angriffsszenarien diskutiert werden soll.

### 6.6.1 Angriffspotential

Um das Angriffspotential zu ermitteln, wird ein Angriffsszenario anhand verschiedener Merkmale analysiert. Hierzu zählen nach der Common Criteria im Beispiel des BSI nach [26] folgende Faktoren:

- Die benötigte Angriffszeit $AP_T$
- Das benötigte Expertenwissen $AP_E$
- Die verfügbaren Informationen $AP_I$

- Der bestehenden Angriffsmöglichkeiten $AP_Z$
- Die verfügbare Ausrüstung $AP_A$

Das Angriffspotential AP kann somit als Summe berechnet werden mit Gleichung 6.1.

$$AP_{Gesamt} = AP_T + AP_E + AP_I + \ AP_Z + AP_A \tag{6.1}$$

Tabelle 6.13 beschreibt die Wertetabelle für die Gewichtungen der Angriffspotentiale.

**Tabelle 6.13:** Wertetabelle für die Gewichtungen der Angriffspotentiale nach [26]

| *Werte* | |
|---|---|
| | *Benötigte Angriffszeit* |
| *< 1Tag* | 0 |
| *< 1 Woche* | 1 |
| *< 2 Wochen* | 2 |
| *< 1 Monat* | 4 |
| *< 2 Monate* | 7 |
| *< 3-6 Monate* | 10-17 |
| *> 6 Monate* | 19 |
| | *Benötigtes Expertenwissen* |
| *Laie* | 0 |
| *Geübte Person* | 3 |
| *Experte* | 7 |
| *Umfassender Experte* | 11 |
| | *Verfügbare Informationen* |
| *Öffentlich* | 0 |
| *Begrenz* | 1 |
| *Sensitiv* | 4 |
| *Streng Vertraulich* | 11 |
| | *Bestehende Angriffsmöglichkeit* |
| *Unbegrenzt* | 0 |
| *Leicht* | 1 |
| *Moderat* | 4 |
| *Schwer* | 10 |
| *Nicht möglich* | 0 |
| | *Benötigte Ausrüstung* |
| *Standard* | 0 |
| *Spezial* | 4 |
| *Maßgeschneidert* | 7 |
| *Vielfach – maßgeschneidert* | 9 |

Für die Gesamtbewertung und Einordnung des Gesamt-Angriffspotentials gibt es nach [115] und [26] einen definierten Wertebereich, der in Tabelle 6.14 dargestellt ist. Prinzipiell bietet ein niedriges Angriffspotential einen geringen Schutz. Somit ergibt sich mit höherem Potential ein höherer Aufwand, um einen Angriff durchzuführen. In Kontradiktion dazu steht die Angriffsdurchführbarkeit, womit ein niedriges Angriffspotential eine hohe Angriffsdurchführbarkeit beschreibt.

**Tabelle 6.14:** Klassifizierung des Angriffspotentials

| Sehr niedrig | 0–9 | Weniger als Niedrig |
|---|---|---|
| Angriffspotential | Wertebereich | Definition nach [26] |
| Niedrig | 10–13 | Der Schutz vor zufälligem, unbeabsichtigtem Eindringen muss gegeben sein. Ein sachkundiger Angreifer hingegen kann diesen Schutz überwinden. |
| Mittel | 14–19 | Der Schutz muss vor einem Angreifer mit beschränkten Gelegenheiten oder Betriebsmitteln gegeben sein. |
| Hoch | 20-24 | Der Schutz darf nur von Angreifern überwunden werden, die über sehr gute Fachkenntnisse sowie Gelegenheiten und Betriebsmittel verfügen. |
| Sehr hoch | >=25 | Mehr als hoch |

### 6.6.2 Schadenspotential

Um das Schadenspotential zu bestimmen, werden drei Arten von Schäden berücksichtigt, die bei einem erfolgreichen Angriff verursacht werden:

- Sicherheitsschaden: Hier entstehen infolge eines erfolgreichen Angriffes Schäden nicht nur am Material, sondern auch an Personen.
- Finanzieller Schaden: Hier entstehen finanzielle Verluste aus einem Business-Modell oder auch rechtliche Konsequenzen (Strafen), Reputationsschäden oder der Verlust von Marktanteilen.

- Betriebs-/Funktionsschäden: Diese Einordnung umfasst Betriebsschäden und alle anderen ungünstigen Vorkommnisse. Dies betrifft beispielsweise Schäden an der Fahrzeugfunktionalität.

Diese Kategorien werden in Tabelle 6.15 nach Leveln unterteilt und einem Schadenspotential zugeordnet.

**Tabelle 6.15:** Schadenskategorien

| Schadenskategorie | Level | Schadensart (Beispiel) | Schadenspotential |
|---|---|---|---|
| Sicherheitsschaden | S0 | Kein Schaden | Kein |
| | S1 | Unerwartetes Verhalten in der Steuerung | Mittel |
| | S2 | Angreifer verfügt über indirekte Ansteuerungsmöglichkeiten | Kritisch |
| | S3 | Angreifer hat Kontrolle | Katastrophal |
| Finanzieller Schaden | F0 | Kein finanzieller Schaden | Kein |
| | F1 | Mittlere finanzielle Schäden, z. B.: hohe Servicekosten | Mittel |
| | F2 | Gefährdung des Business Case | Kritisch |
| Betriebs-/ Funktionsschäden | B0 | Keine Effekte | Kein |
| | B1 | Unbedeutende funktionale Änderungen | Unbedeutend |
| | B2 | Komforteigenschaften beeinflusst | Mittel |
| | B3 | Kompletter Ausfall | Kritisch |

### 6.6.3 Risiko

Nach der Ermittlung und Klassifizierung der Angriffs- und Schadenspotentiale kann nun eine Risikomatrix erstellt werden. Tabelle 6.16 zeigt die allgemeinen Bewertungskriterien nach [26], wobei die Angriffspotentiale in Zeilen- und die Schadenspotentiale in Spaltenform dargestellt sind.

**Tabelle 6.16:** Risikomatrix

| **Risiko** | | **Schadenspotential** | | | | |
|---|---|---|---|---|---|---|
| | | Kein | Unbedeutend | Mittel | Kritisch | Katastrophal |
| **Angriffspotential** | Sehr hoch | Kein | Kein | Kein | Gering | Gering |
| | Hoch | Kein | Kein | Gering | Erhöht | Erhöht |
| | Mittel | Kein | Gering | Erhöht | Erhöht | Mittel |
| | Niedrig | Kein | Gering | Erhöht | Mittel | Hoch |
| | Sehr niedrig | Kein | Gering | Mittel | Hoch | Hoch |

### 6.6.4 Angriffs- und Schadenspotential

In diesem Abschnitt wird das Angriffs- und Schadenspotential jedes Angriffsszenarios analysiert. Eine exemplarische Form der Angriffsszenarien für das Sicherheitsobjekt der sicheren Laufzeitumgebung ist in einer tabellarischen Übersicht in Tabelle 6.17 dargestellt. Die gesamte Risikoanalyse befindet sich im Anhang.

**Tabelle 6.17:** Risikoanalyse

| Sicherheitsobjekt | | | | | Angriffspotential | | | | | | | Schadenspotential | Risiko |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Objektname | # | Angriffsszenario | Modell | Sicherheitsrelevante Systemkomponenten | $AP_{Zeit}$ | $AP_{Expertise}$ | $AP_{Information}$ | $AP_{Angriff}$ | $AP_{Ausrüstung}$ | AP | AP_Name | | |
| Sichere Laufzeitumgebung | 1 | Manipuliertes Register | Baseline | ROOT.SCHLÜSSEL | 0 | 7 | 11 | 4 | 7 | 29 | Sehr hoch | Katastrophal | Gering |
| | | | | ROOT.SCHLÜSSEL | 0 | 7 | 11 | 4 | 7 | 29 | Sehr hoch | Katastrophal | Gering |
| | 2 | Manipulierte Applikation | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 7 | 4 | 4 | 22 | Hoch | Katastrophal | Erhöht |
| | | | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| | | | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| | 3 | Manipulierte Datensätze | Baseline | Baseline.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | Modell_1 | Modell_1.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | Modell_2 | Modell_2.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | 4 | Injizierte Datensätze | Baseline | Baseline.ROOT.INDEX_DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Mittel | Gering |
| | | | | Baseline.ROOT.INDEX_ROOT.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | Modell_1 | Modell_1.ROOT.INDEX_STRUKTUR.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | Baseline | V2.STRUKTUR.INDEX_SCHLÜSSEL.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | | V2.STRUKTUR.INDEX_DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Mittel | Gering |
| | | | Modell_1 | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | Modell_2 | Modell_2.ROOT.INDEX_STRUKTUR | 0 | 7 | 1 | 4 | 4 | 16 | Mittel | Kritisch | Erhöht |
| | | | | Modell_2.STRUKTUR.INDEX_SCHLÜSSEL | 0 | 7 | 1 | 4 | 4 | 16 | Mittel | Kritisch | Erhöht |
| | | | | Modell_2.STRUKTUR.INDEX_DATEN | 0 | 7 | 1 | 4 | 4 | 16 | Mittel | Mittel | Gering |
| | | | | Modell_2.STRUKTUR.INDEX_ID.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| | | | | Modell_2.SCHLÜSSEL.INDEX_BLOCK | 0 | 7 | 1 | 4 | 4 | 16 | Mittel | Kritisch | Erhöht |

Aus Kapitel 6.5 sind die Sicherheitsobjekte mit zugehörigen Angriffsszenarien aus 6.6 und zugeordneten sicherheitsrelevanten Systemkomponenten gelistet. Das Risiko wurde nach den Angriffspotentialen aufgeschlüsselt und mit dem Schadenspotential ermittelt. Die Einstufungen des Schadenspotentials ergeben sich aus sicherheitskritischen Funktionen. Durch einen manipulierten Root-Schlüssel könnte das gesamte System beeinträchtigt werden und das kann auch Auswirkungen auf sicherheitsrelevante Funktionen haben. Weiterhin „Kritisch“, jedoch nicht als „Katastrophal“ betrachtet werden statische Datensätze. Da innerhalb der auszuführenden Applikation Plausibilitätsprüfungen vorausgesetzt werden. Final ergibt sich das Risiko aus der Korrelation zwischen dem Schadenspotential und dem Gesamtwert des Angriffspotentiales AP. Die Angriffspotentiale wurden durch

eine Expertengruppe erhoben. Im Anschluss werden in Kapitel 6.8 die Angriffsszenarien erläutert.

## 6.7 Evaluierung der Risikoanalyse

Nachdem die Risikoanalyse erstellt worden ist, können anhand der sicherheitsrelevanten Systemkomponenten die Potentiale und Gefahren der verschiedenen Modelle evaluiert werden. Der Fokus auf die entstandenen Risiken der Komponenten liegt im Angriffspotential.

Im folgenden Abschnitt werden die drei verschiedenen Modelle anhand der gegebenen vier Sicherheitsobjekte miteinander verglichen.

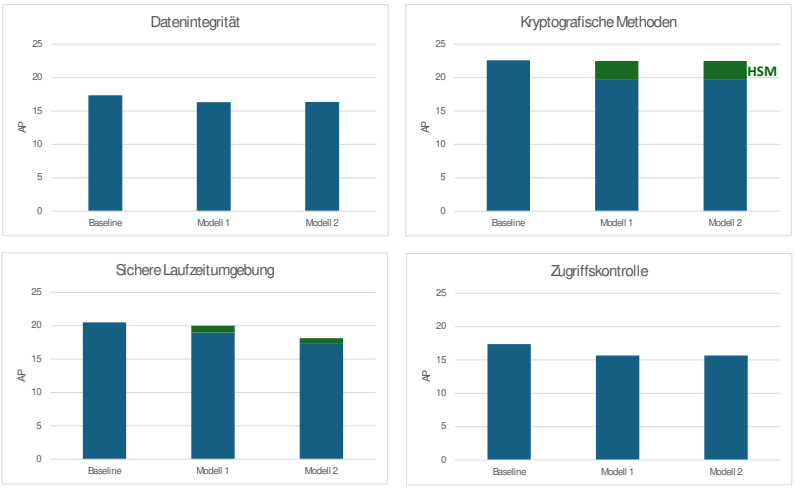


**Abbildung 6.7:** Evaluierung der AP-Mittelwerte

Abbildung 6.7 wertet anhand des resultierenden Angriffspotentials AP die vier Sicherheitsobjekte im Hinblick auf die verschiedenen Modelle aus. Insbesondere hinsichtlich der kryptografischen Methoden ist das Potential im Modell der Baseline am größten, da spezifische Hardwarekomponenten der Root-Instanz kompromittiert werden müssten. Daher wird im Folgenden das Modell *Baseline* als Referenzmodell betrachtet, da ausschließlich eine Autorität (Root-Instanz), verwendet wird. Weiterführend sind zwei Annahmen getroffen, ob Modell_1 und Modell_2 in Rahmen des Validierungsprozesses auf kryptografische Berechnungsfunktionen eines Hardware-Security-Modules zugreifen können. Diese Differenz ist als HSM-Differenz hintelegt.

Durch den Einsatz Instanz-orientierter Autoritäten und einer sequentiellen Bootabfolge unterscheidet sich der Mittelwert des Angriffspotentiales AP hinsichtlich der Datenintegrität nur minimal um einen AP Punkt und liegt somit in der gleichen Dimensionierung bei 17, bzw. 16 AP Punkten.

Kryptografischen Methoden in Modell_1 und Modell_2 differenzieren sich im Vergleich zum Referenzmodell anhand des Einsatzes eines HSM. Hier liegt die Differenz ohne Einsatz bei knapp drei AP Punkten, mit Einsatz jedoch verhält sich das Angriffspotential jedoch gleichwertig. Somit weisen die Modelle gleichwertig ein Potential bei 22 AP Punkten auf.

Die Modellierung in Modell_2 hat die Eigenschaft, dass die Indizes nicht im Signierungsprozess enthalten sind. Dies ist insbesondere hinsichtlich der sicheren Laufzeitumgebung zu erkennen. So ist das Potential des Modell_2 zum Referenzmodell (21 AP Punkte) um drei AP Punkte geringer.

Gleichwohl ist im Rahmen des Sicherheitsobjektes der Zugriffskontrolle die Differenz nur minimal. So ist das Potential der Modelle Modell_1 und Modell_2 zum Referenzmodell (17 AP Punkte) im Gesamtpotential

gleichwertig 2 AP Punkte geringer.

### 6.7.1 Datenintegrität

Der Authentifizierungs-Algorithmus kann schlecht ausgelegt werden und ermöglicht verschiedene Angriffsszenarien. Diese sind häufige Ursachen für Sicherheitslücken. Sie umfassen z. B. keine Einschränkung falscher Authentifizierungsversuche. Daher wurde das Risiko durchgehend mit „Mittel“ bis „Erhöht“ eingeschätzt.

#### 6.7.1.1 Nicht authentifizierte Modifikation der Datensätze

Dieser Angriff ist aus Sicht des Schadenspotentials kritisch, in Teilen katastrophal, wenn Schlüssel betroffen sind. Um Änderungen wahrzunehmen, sollten entsprechende Maßnahmen implementiert werden. Das Einbringen eines manipulierten Zertifikats oder eines Codeblockes kann bei unzureichenden kryptografischen Maßnahmen Softwarepakete frei signieren, siehe „nicht konformer kryptografischer Schlüssel“. Unter der Annahme, dass hier die Zugriffsmöglichkeiten und die dafür benötigte Ausrüstung (wie z. B. ein mechanischer Nadelkontaktieradapter) wie auch ein Expertenwissen erforderlich sind, wird das Risiko als „Erhöht“ eingestuft.

#### 6.7.1.2 Tausch des Speicherbausteines

Diese Attacke ist vergleichbar mit dem Angriff „Nicht authentifizierte Modifikation der Datensätze“ und besitzt die Risikostufe „Erhöht“. Speicherchips (oder sogar Prozessoren mit einem Chipspeicher) sind in der Regel keine Sonder-ASICs und im Fachhandel erhältlich. Hierzu bedarf es zusätzlicher Ausrüstung, wie einer BGA-Lötmaschine bei einem Komponententausch. Die Verzögerung dieses Angriffes bildet der Lötprozess, was auf einer komplexen Leiterplatte ziemlich schwierig sein kann, zumal sich das Gerät nicht im eingebauten Zustand und somit außerhalb des Systems befindet. Allerdings ist der mechanische Eingriff wiederum durch eine geübte Person anwendbar. Abhängig vom Layout der Leiterplatte und von den ausgewählten Hardwarekomponenten ist diese Option möglicherweise einfacher als das Angriffsszenario „Nicht authentifizierte Modifikation der Datensätze“. Durch den ausgebauten Zustand hingegen ist die Zugriffsgelegenheit sehr eingeschränkt und bietet eine aufwendige Angriffsmöglichkeit.

#### 6.7.1.3 Tausch des Gerätes

Komplexer ist der Tausch des gesamten Gerätes bzw. der Tausch des Controllers zum Manipulieren der Root-Instanz. Unter Beibehaltung des Angriffsszenarios zum Tausch des Speicherbausteins minimiert sich das Risiko durch den Aspekt, dass hier Expertenwissen erforderlich ist, auf „Gering“.

#### 6.7.1.4 Unvollständiges SW-Update

Dieses Szenario ist vergleichbar mit der nicht authentifizierten Modifikation der Datensätze im Speicherbaustein und wird mit dem Risiko „Erhöht“ bis „Mittel“ eingestuft.

### 6.7.2 Kryptografische Methoden

Im Folgenden werden die Angriffsszenarien der kryptografischen Methoden vorgestellt.

#### 6.7.2.1 Schwachstelle in der Rechnerarchitektur

Analog zu den „Architekturbezogenen Sicherheitslücken“ besteht die Möglichkeit, dass eine ausnutzbare Sicherheitsanfälligkeit in einem System gefunden wird. Es ist anzunehmen, dass Skripte oder Schadsoftware frei verfügbar sind, da die Information frei zugänglich ist. Entsprechend könnte der Angriff durch eine geübte Person erfolgen. Werkzeuge und bekannte Schwachstellen von gängigen Bibliotheken werden ebenfalls ausgenutzt. Daher wird das Schadenspotential als „Katastrophal“ bewertet. Gleichwohl wird in Betracht gezogen, dass weitere Abhängigkeiten, wie z. B. zusätzliche Authentifizierungsmaßnahmen etc., ebenfalls überwunden werden müssen, womit sich ein Risiko in der Einstufung „Erhöht“ ergibt.

Sollte wie in Abbildung 6.7 es nicht zu einem Gebrauch eines HSM im Modell_1 und Modell_2 kommen, würde sich das Angriffspotential auf „Mittel“ verringern und es somit zu einem erhöhten Gesamtrisiko auf

„Mittel“ auswirken.

### 6.7.3 Sichere Laufzeitumgebung

Bei diesem Angriffsszenario wird davon ausgegangen, dass die Attacke für die nicht authentifizierte Modifikation der Datensätze erfolgreich war und nun beabsichtigt wird, eine geänderte (oder andere) Operation auszuführen. Daher wird das Schadenspotential weitestgehend als kritisch eingestuft.

Im Kern wird zwischen den drei Varianten der Manipulation der Register in der Root-Instanz, der Applikationen, Daten-Blöcke und sämtlicher Indizes unterschieden. Die Root-Instanz bildet hier einen Sonderfall, da die manipulierten Register in der Root-Ebene verortet sind.

#### 6.7.3.1 Manipuliertes Register

Für die aktuellen Architekturen sind sichere Startmechanismen geplant, womit die Expertise, aber auch die benötigte Information entscheidend ist und entsprechend die Angriffspotentiale stark ausgebildet.

#### 6.7.3.2 Manipulierte Applikation

Dieses Angriffsszenario ähnelt der „Nicht authentifizierten Modifikation der Datensätze“. Der Angreifer ändert/manipuliert eine vorhandene Applikation und führt sie aus, während in diesem Fall in der Laufzeitumgebung auch neue Anwendungen ausgeführt werden können. Daher wird

das Schadenspotential als „Katastrophal“ eingestuft. Es erfordert jedoch in Teilen sensitive Informationen und Expertenwissen, sodass das Angriffspotential gesteigert und somit das Risiko auf „Mittel“ erhöht wird. Eine Ausnahme bildet hier das Referenzmodell , da durch die Verwendung des Root-Schlüssels auch die zugehörige Informationsbeschaffung in die Gewichtung fällt.

#### 6.7.3.3 Manipulierte Datensätze

Die gleiche Einschätzung trifft auch beim Angriffsszenario für die manipulierten Datensätze zu. Dadurch, dass von einer Plausibilitätsprüfung in der Applikation auszugehen ist, wird das Schadenspotential mit „Kritisch“ eingeschätzt. Das resultierende Risiko wird entsprechend bei „Erhöht“ verortet.

#### 6.7.3.4 Injizierte Datensätze

Für die injizierten Datensätze der Indizes liegt die Kernbetrachtung im Verhalten des Angriffspotentials. Weil das Modell_2 Indizes verwendet, die nicht dem Signierungsprozess unterliegen, besteht die Gefahr, dass eine Manipulation dieser Datensätze in einem der nicht signierten Indizes ein niedriger Aufwand und somit ein geringeres Angriffspotential und ein erhöhtes Risiko bietet. Die Evaluierung zeigt nun, dass sich bezüglich des Angriffspotentials der bestehenden Angriffsmöglichkeit ein leichterer Zugriff ergibt.

Im Resultat bedeutet dies, dass sich aufgrund des Ausschlusses des Signierungsprozesses das resultierende Angriffspotential gegenüber dem Referenzmodell verändert. Ursache hierfür ist, dass sämtliche Datensätze und Applikationen, öffentliche Schlüssel und Indexblöcke weiterhin signiert sind und sich dies aufgrund einer solchen Konstellation im Schadensfall in Form eines Fehlers im Sinne einer Plausibilität oder beim Verifizierungsprozess niederschlägt.

### 6.7.4 Zugriffskontrolle

Das Sicherheitsobjekt der Zugriffskontrolle ergibt sich aus folgenden vier Angriffsszenarien:

#### 6.7.4.1 Brute-Force-Angriff

Ein schwacher Authentifizierungs-Algorithmus oder eine schlecht entworfene Authentifizierungs-Schnittstelle ermöglichen einen erfolgreichen (automatisierten) Brute-Force-Angriff. Die Kernaspekte hierfür sind detektierende Maßnahmen im Rahmen fehlerhafter Authentifizierungsversuche. Prinzipiell wäre ein Laie als Angreifer ausreichend, um solch einen Angriff zu verüben, ist jedoch wegen eines sehr großen Zeitrahmens und starken Authentifizierungs-Algorithmus vom Risiko her vernachlässigbar. Daher ist die Einschätzung eines erfolgreichen Brute-Force-Angriffs durch einen Experten zu betrachten. In Teilen der auf die Rechnerarchitektur bezogenen Komponenten fungiert er als umfassender Experte. Ein solcher Angriff ist mit standardisierter Ausrüstung zu bewältigen. In jedem Fall wäre jedoch das Schadenspotential katastrophal, weshalb auch das Risiko

als „Hoch“ eingestuft wird.

#### 6.7.4.2 Unautorisierte Übernahme einer existierenden Anmeldung

Für unbefugte Übernahmen einer bestehenden Sitzung ist eine kabelgebundene oder drahtlose Verbindung nötig. Die Voraussetzung für dieses mögliche Szenario ist eine für vorgesehene Software- oder ein zusätzliches Hardwaremodul, dies jedoch in dieser Bewertung als standardisierte Ausrüstung eingestuft ist. Ein verschlüsselter Kommunikationskanal erschwert die Möglichkeit eines solchen Angriffes. Das Angriffspotential dieses Angriffs wurde mit „Erhöht“ bis „Mittel“ bewertet.

#### 6.7.4.3 Nicht konformer kryptografischer Schlüssel

Bei dieser Attacke wird davon ausgegangen, dass der Angreifer eine Methode zum Manipulieren der Daten besitzt. In diesem Fall kann der Angreifer die kryptografischen Schlüssel manipulieren oder lesen. Dies ist eine kritische Handlung, die einen katastrophalen Schaden verursachen kann. Wenn ein symmetrischer oder asymmetrischer Schlüssel ausgetauscht wird, kann die Authentizität gefälscht werden und sich der Eindringling als vertrauenswürdiges Backend ausgeben. Eine manipulative Software kann dann als korrekt authentifiziert werden. Die Beurteilung wurde hier grundsätzlich mit „Erhöht“ bis „Mittel“ eingeschätzt.

#### 6.7.4.4 Offene Schnittstellen

Bei diesem Angriff wird davon ausgegangen, dass eine JTAG- oder UART-Schnittstelle aktiviert und für eine Ansteuerung verfügbar gemacht wird. Diese Schnittstellen sind regelmäßig während der Entwicklung in Gebrauch (z. B. für Debugging, UART-Konsolen usw.), werden jedoch im Rahmen des EOL-Testes verschlossen. Der Angriff ermöglicht eine signifikante Störung der Software und Hardware des Prozessors, wenn der Programmcode nicht anders ausgelegt ist. Daher wird das resultierende Risiko mit „Hoch“ eingeschätzt. Die Ausnahme bilden Schnittstellen, die eine Einsprungbedingung (Pattern) und zusätzliche Authentifizierung erfordern, die Verwendung findet und nach dem Bootvorgang wieder deaktiviert wird. Das Risiko dieses Angriffs wird als „Mittel“ definiert, da hier andere Sicherheitsmodule greifen.

## 6.8 Sicherheitsanforderungen

Dieser Abschnitt enthält eine Aufstellung der resultierenden Sicherheitsanforderungen, die auf der Grundlage der Risikoanalyse und ihrer Evaluierung abgeleitet wurde.

### 6.8.1 Anforderungen für die Zugriffskontrolle

Die Authentifizierungs-/Autorisierungs-Algorithmen müssen einen ausreichend langen kryptografischen Schlüssel oder eine entsprechende Kennwortlänge verwenden. Am Beispiel der i.MX6-Plattform [16, 109] sollten

verwendete Schlüssel eine Länge von 4096 Bit aufweisen. Dies würde die Angriffszeit erheblich vergrößern und somit ein höheres Angriffspotential ergeben. Hier muss auch bei Anmeldeversuchen ein Fehlerzähler implementiert und bei Wiederholungsprüfungen eine Zeitpause eingeräumt werden. Auch für unautorisierte Übernahmemaßnahmen ist es sinnvoll, den Kommunikationsweg zu verschlüsseln.

### 6.8.2 Anforderungen für die Datenintegrität

Einen Austausch ganzer Speicher-ICs muss die Plattform erkennen und nichtauthentifizierte Änderungen von Hardware oder Manipulation von Softwaremodulen erfassen. Dennoch sollte das System gerade hinsichtlich des „Design for Repair“ Reparaturprozesse wie das Remanufacturing ermöglichen. Hierfür bilden die Authentifizierungsmaßnahmen Autoritäten und eine Grundlage, um beide Anforderungen miteinander zu kombinieren. Gerade hinsichtlich der mehrstufigen Integritätsprüfung des Codeblockes und der Autorität der Zielapplikation werden Änderungen in der Laufzeitumgebung festgestellt.

### 6.8.3 Anforderungen für kryptografische Methoden

Die Plattform muss bei Markteinführung öffentliche, standardisierte und aktuelle kryptografische Algorithmen und Protokolle verwenden (z. B. AES, RSA, TLS). Eingehend evaluierte kryptografische Algorithmen (siehe Empfehlungen von BSI, NIST oder ENISA) werden bevorzugt.

Hierbei ist eine detaillierte Validierung des Designs der eingesetzten kryptografischen Protokolle und Algorithmen durch unabhängige Überprüfungen durchzuführen. Wie bei einem gängigen Software-Entstehungsprozess und dessen Freigabeerprobung ist eine statische und dynamische Codeanalyse der Implementierung durchzuführen. Diese Zielsetzung beinhaltet den Schutz vor Seitenkanalangriffen in Form von seitenkanalresistenten Hardwaresicherheitsmodulen.

Hier äußert sich der Vorteil in den Modellen Modell_1 und Modell_2 darin, dass während der Laufzeit im Rahmen eines Softwareupdates auch die kryptografischen Methoden angepasst werden können und somit im schlimmsten Fall einer kompromittierten, kryptografischen Methodik eine Obsoleszenz des Gerätes verhindert werden kann. Gleichwohl ist die Verwendung eines HSM Modules geboten, da dies das Angriffspotential signifikant erhöht.

### 6.8.4 Anforderungen für die sichere Laufzeitumgebung

Es muss in der Applikation durch Plausibilitätsprüfungen sichergestellt sein, dass es zu keinem unerwarteten Fehler kommen kann, wenn ein Index manipuliert wurde. Gleichwohl soll durch die sichere Ausführungsumgebung einer Applikation garantiert werden, dass auch Schreibrechte und der Zugriff auf die affektierten Adressbereiche nur erlaubten Instanzen ermöglicht werden.

## 6.9 Fazit

Dieses Kapitel behandelt die Evaluierung der Realisierung zur Updatefähigkeit von irreversibel geschützten Applikationen mit neuer Autorität. Die Updatefähigkeit einer Zielapplikation mit den Modellierungen Modell_1 und Modell_2 zur Verbesserung des Produktlebenszyklus wurde in Relation zum Referenzmodell betrachtet.

Das in diesem Kapitel erstellte Sicherheitskonzept gewährt einen wesentlichen Einblick in die Herausforderungen der Realisierung der festgestellten Anforderungen. Zunächst ist unter Betrachtung der Neuausrichtung des Modells hierbei die Öffnung einer Schnittstelle im System kritisch zu bewerten.

Die Analyse in Form des Angriffs- und Schadenspotentials hat ergeben, dass dennoch kein erhöhtes Sicherheitsrisiko resultiert. Die Ursache liegt im Ansatz des Verheiratungsprozesses der irreversibel gesetzten Autorität zu der ersten Instanz.

# 7 Zusammenfassung und Ausblick

Dieses Kapitel beinhaltet die Schlussfolgerung der in Kapitel 4 formulierten Forschungsfragen. Im Hinblick auf die Umsetzung für die Updatefähigkeit neu signierter Zielapplikationen und deren Validierung wird im ersten Abschnitt des Kapitels 7 eine inhaltliche Zusammenfassung gegeben. Der zweite Abschnitt resümiert den Forschungsbeitrag mit dem Abgleich der im dritten Abschnitt zusammengefassten Forschungsfragen aus Kapitel 4. Weiterführend wird im vierten Abschnitt eine Abgrenzung beschrieben und final im fünften ein Ausblick geboten.

## 7.1 Inhalt

Im Rahmen der Erarbeitung der in Kapitel 4 definierten Fragestellungen wurde ein Verfahren entwickelt, das die Befähigung zum Betreiben mehrerer Applikationen mit unterschiedlichen Autoritäten sicherstellt. Hierbei liegt der Fokus auf der Updatefähigkeit neu signierter Zielapplikationen.

Beispiele für derartige dynamische Systeme, die dieses Verfahren zur Verwendung bringen können, sind, wie im Leitbeispiel angefügt, Steuergeräte

aus dem Infotainment-Bereich, ebenso auch fahrdynamische Systeme (etwa für die als elektronisches Stabilitätsprogramm [ESP] bekannte Fahrdynamikregelung) sowie Steuergeräte für das zumindest teilweise automatisierte Steuern von Fahrzeugen im Verkehr. Dies betrifft ebenfalls künftige Generationen von zentralisierten Steuerungsgeräten.

Im Rahmen der Erarbeitung des Verfahrens wurden insgesamt neun Patentanmeldungen eingereicht, die in [116, 117, 118, 119, 120, 121, 122, 123, 124] aufgelistet sind.

Darin wird für das im Kapitel 5 erstellte Basismodell folgende Definition festgehalten, wonach unter einer Applikation eine nominelle Funktionalität des Steuergerätes implementiert ist. Sie umfasst mindestens einen Codeblock, der einen ausführbaren Programmcode enthält. Neben dem Codeblock einer Applikation kann auch das System mindestens einen Datenblock einschließen, den der Programmcode zur Erfüllung seiner Funktion benötigt. Der Datenblock kann beispielsweise Kennfelder oder andere Parameter wie etwa ein trainiertes Gewicht enthalten, das ein Machine-Learning-Modul charakterisiert.

Im Rahmen des Verfahrens wird anhand eines in dem Steuergerät fest hinterlegten Instanz-orientierten Layouts ein erster Codeblock einer ersten Applikation, der einen ausführbaren Programmcode dieser Applikation enthält, abgerufen.

### 7.1.1 Secure Boot

Anhand von in dem Steuergerät irreversibel hinterlegten Verifizierungsinformationen in der Root-Instanz wird geprüft, ob dieser erste Codeblock auf

einem zu diesen Verifizierungsinformationen korrespondierenden Stand ist. Dieses irreversible Element kann ohne Wissen und Wollen des Herstellers des Steuergerätes oder einer anderen zur Vornahme von Modifizierungen am Steuergerät autorisierten Stelle nicht frei verändert werden. Hierfür können beispielsweise Speichermodule oder Speichermedien in Form von Fuse-Registern oder Speicherbereichen verwendet werden, in denen einzelne Bits, Bytes oder Blöcke von Bytes ausgehend vom Neuzustand durch einen irreversiblen physikalischen Schreibprozess auf neue Werte gesetzt werden können. Ein Beispiel hierfür sind Speichermodule, in denen einzelne Bits infolge des Durchbrennens oder anderweitigen Zerstörens elektrisch leitender Strukturen umgeschaltet werden. Als Antwort darauf, dass das Ergebnis dieser Prüfung positiv ist, wird der im ersten Codeblock (Root-Block der ersten Instanz) enthaltene Programmcode zur Ausführung freigegeben und ausgeführt.

### 7.1.2 Lösungskonzept

Durch einen in dem ersten Root-Block enthaltenen Index wird der Interne-Struktur-Block abgerufen, der einen Verweis in Form von Indizes auf einen weiteren Codeblock für einen Schlüssel- oder Datenblock führt. Anhand von im ersten Root-Block hinterlegten Verifizierungsinformationen respektive den kryptografischen öffentlichen Schlüsseln wird geprüft, ob der Interne-Struktur-Block auf einem zu diesen Verifizierungsinformationen (Signatur) korrespondierenden Stand ist. Als Antwort darauf, dass das Ergebnis dieser Prüfung positiv ausfällt, wird anhand der in dem Interne-Struktur-Block enthaltenen Indizes der weitere Codeblock dieser Instanz abgerufen.

In der gleichen Weise enthält auch der Schlüssel-Block einen Verweis für eine weitere Instanz und somit die neue Applikation. Hinzu kommt eine hinterlegte Verifizierungsinformation respektive der kryptografische öffentliche Schlüssel, um zu prüfen, ob der Root-Block der neuen Instanz dem geforderten, korrespondierenden Stand entspricht. Im Sonderfall gilt aber auch, dass ein nächster Datenblock mit eigener korrespondierender Autorität abgerufen werden kann.

### 7.1.3 Risikoanalyse

Mit dem Verfahren kann der Bootvorgang eines Steuergerätes über die Kernanforderung, nur einen autorisierten Code auszuführen, dahingehend abgesichert werden, dass für die Reihenfolge der realisierten Applikationen und somit Instanzen ein Zwangsweg vorgegeben wird. Für den Endnutzer des Steuergerätes ist es nicht möglich, diese Reihenfolge eigenmächtig zu ändern oder die Ausführung bestimmter Applikationen selbständig zu unterdrücken.

Das bedeutet im Umkehrschluss, dass die Zergliederung der Funktionalität des Steuergerätes in einzelne Applikationen, die nacheinander auszuführen sind, kein besonderes Sicherheitsrisiko mehr darstellt.

Hierzu wurde in der Risikoanalyse aus Kapitel 5 die Erkenntnis gewonnen, dass eine monolithische Applikation, die die gesamte Funktionalität des Steuergerätes mittels der Root-Autorität abdeckt, sich zwar besonders gut gegen Manipulation absichern lässt, im Gegenzug jedoch vergleichsweise umständlich upzudaten bzw. zu erweitern ist.

Insbesondere bei Steuergeräten für Fahrzeuge, die ein breites Spektrum an Aufgaben abdecken, sind die jeweiligen Applikationen häufig von mehreren Parteien entwickelt worden. Ein weiterer Aspekt ist, dass, wenn am Ende eine monolithische Applikation entstehen muss, ein Hersteller die Beiträge von allen anderen Herstellern zusammenführen und diese bei jedem Update einer Einzelkomponente wiederholen muss. Sind hingegen die von verschiedenen Herstellern implementierten Funktionalitäten in mehreren Applikationen untergebracht, können Letztere unabhängig voneinander upgedatet werden, was eine Service-orientierte Bereitstellung neuer Softwarekomponenten unterstützt.

Jedoch schafft dies ohne die hier beschriebene zusätzliche Sicherung Möglichkeiten, in den Bootvorgang des Steuergerätes einzugreifen.

## 7.2 Forschungsbeitrag

Der Forschungsbeitrag zeichnet sich durch die in Kapitel 4 abgeleiteten Forschungsfragen aus. Die Kernmotivation zur Erarbeitung der Updatefähigkeit einer Zielapplikation mit neuer, gültiger Autorität ergibt sich aus dem Aspekt zur Erweiterung des Produktlebenszyklus. Hierzu ist der Re-Use-Aspekt im Fokus, um den Lebenszyklus signifikant zu erhöhen und moderne Nachserienversorgungsstrategien ebenso zu verbessern.

Jedoch auch in der Laufzeit des Produktes selbst ist der Gesichtspunkt zur Verbesserung und Erneuerung von Updatefähigkeiten mit neuer, gültiger Autorität für eine Service-orientierte Bereitstellung aktueller Softwarekomponenten ein tragendes Element hinsichtlich des Forschungsbeitrages.

Die entwickelte Methodik lässt sich auch auf andere Gebiete ausweiten, etwa Produktreihen, die für den Einsatz in vielen Ländern hergestellt werden. Durch verschiedene, regulatorische Vorgaben in den einzelnen Ländern unterliegt die Nutzung des Produktes unterschiedlichen Eigenschaften. Rein technisch sind jedoch diese Produktreihen mit allen Softwarekomponenten als Funktionen ausgerüstet, die in irgendeinem der vorgesehenen Länder gebraucht werden, beispielsweise zur Auslegung der maximal benötigten Leistung und Geschwindigkeit im jeweiligen Land. Die Einhaltung dieser Bestimmungen ist eine zwingende Voraussetzung dafür, dass das Produkt so benutzt werden darf und beispielsweise eine Freigabe oder Abnahme einer zuständigen Behörde erhält.

Im Bootprozess des Produktes respektive Systems wird demgemäß im Root-Block einer Instanz eine Applikation sitzen, die je nach Einsatzland den Rahmen des Erlaubten festlegt. Eine Applikation in einer Folgeinstanz, beispielsweise für ein Antriebssystem, lässt sich entsprechend den so festgelegten Parametern tatsächlich einsetzen. Wenn es einem Angreifer nun gelingt, die Bootreihenfolge umzudrehen, kann er auch die nicht zulässigen Funktionen nutzen sowie mit der technisch maximal möglichen Geschwindigkeit fahren und nachfolgend mit einem nicht zugelassenen Zustand das Produktes betreiben.

Anhand des im Interne-Struktur-Block referenzierten Schlüsselblockes und der darin enthaltenen Verifizierungsinformationen wird geprüft, ob die nächste Instanz auf einem zu diesen Verifizierungsinformationen korrespondierenden Stand ist. Falls das Ergebnis dieser Prüfung positiv ausfällt, wird der im neuen Root-Block enthaltene Programmcode zur Ausführung freigegeben. Auf diese Weise kann, ausgehend von einem einzigen im Steuergerät fest hinterlegten Verweis und einem singulär dort fest und

irreversibel hinterlegten Datensatz, mit Verifizierungsinformationen (Vertrauensanker, „Root of Trust“) eine Vertrauenskette („Chain of Trust“) aufgebaut werden, die über eine beliebige Anzahl nacheinander auszuführender Applikationen reicht.

Durch diese Ausgestaltung wird anhand der in dem ersten Root-Block enthaltenen Verweise nicht unmittelbar der Schlüsselblock beschafft, sondern zunächst der Interne-Struktur-Block. Mit Hilfe der im Root-Block hinterlegten Verifizierungsinformationen wird dann geprüft, ob der Interne-Struktur-Block sich auf einem zu diesen Verifizierungsinformationen korrespondierenden Stand befindet. Sofern das Ergebnis dieser Prüfung positiv ist, wird mittels eines in dem Interne-Struktur-Block enthaltenen Verweises der Schlüsselblock beschafft. Da es eine Vertrauenskette von den Verifizierungsinformationen (öffentlicher Schlüssel) im Root-Block zu der eigenen Instanz gibt, ist es für das Sicherheitsniveau unerheblich, mit welchen Verifizierungsinformationen der Datenblock bestätigt wird. Wenn der Datenblock eigene Verifizierungsinformationen enthält, schafft dies eine noch größere Granularität dahingehend, welche Entitäten den Bootablauf des Steuergerätes in welcher Weise abändern können.

Das Zwischenschalten eines Interne-Struktur-Blockes erleichtert insbesondere die Aufgabentrennung, beispielsweise in dem eingangs beschriebenen Szenario, in dem Applikationen von verschiedenen Herstellern Instanzorientiert zusammenwirken. So kann beispielsweise mittels des Interne-Struktur-Blockes ausgewählt werden, welche Applikation als Nächstes ausgeführt wird, gleichwohl jedoch die Wartung und Updates der Nachfolgeinstanz der jeweiligen dritten Partei überlassen werden.

Somit lassen sich auch Zuständigkeiten und die rechtlichen Vorgaben bezüglich der Benutzung des Produktes in entsprechende Betriebsparameter umsetzen. Der Hersteller des Produktes ist hingegen für die Produkthaftung

verantwortlich, ein Projektteam ggf. für die Realisierung einzelner Treiberstufen. Es genügt, unter den im Schlüsselblock gespeicherten Verweisen eine neue Version mit einer digitalen Signatur zu hinterlegen, die sich mit dem im Schlüsselblock hinterlegten öffentlichen Schlüssel bestätigen lässt.

Einen öffentlichen Schlüssel als Verifizierungsinformation zu nutzen, hat den besonderen Vorteil, dass dieser Schlüssel auch dann gleichbleiben kann, wenn der zu verifizierende Root- oder Datenblock geändert wird. Wenn sich die Signatur bestätigen lässt, dann steht fest, dass sie mit dem richtigen geheimen Schlüssel vorgenommen und von einer hierzu befugten Entität geändert wurde. Demgemäß lässt sich durch eine Manipulation des Datenblocks genauso schwerwiegend in die Funktionalität des Steuergerätes eingreifen wie mittels einer Manipulation des ausführbaren Programmcodes. Dabei ist eine zielgerichtete Manipulation beispielsweise durch den Nutzer eines Fahrzeugs an dem Datenblock mitunter noch deutlich einfacher möglich als am Programmcode, den der Nutzer ohne Spezialkenntnisse nicht verstehen kann.

Die öffentlichen Schlüssel als Verifizierungsinformationen kann besonders einfach abgestuft verwalten, wer zur Änderung bestimmter Abläufe und Einstellungen berechtigt ist. So können beispielsweise bestimmte Berechtigungen für den Zugriff auf Systemressourcen des Steuergerätes und/oder für die Ausführung von Aktionen im oder mit dem Steuergerät für eine Änderung lediglich einem Fahrzeughersteller vorbehalten sein.

Durch die Hinterlegung der Berechtigung in einem Index, der sich in einem signierten Interne-Struktur-Block befindet, lässt sich auch der Zugriff der weiteren Applikation auf Systemressourcen bestimmen. Ferner wird auch die Ausführung von Aktionen durch eine weitere Applikation gemäß diesen Berechtigungen begrenzt, insbesondere da eine Angabe von

Berechtigungen im Kontext einer Vorgängerapplikation im Kern Berechtigungen nur jeweils maximal in dem Umfang einbringt, in dem auch die Vorgängerapplikation diese besitzt. Dies erschwert es, durch das Ausführen einer weiteren vorhandenen Applikation, die möglicherweise sogar gültig signiert ist, Berechtigungseinschränkungen zu unterlaufen. Vorteilhaft ist die Angabe von Berechtigungen bezüglich des Zugriffs speziell auf Systemressourcen, damit sich beispielsweise ein Fahrzeug mit Steuergerät nach dem Hochfahren der Dienste immer in einem definierten Zustand befindet.

## 7.3 Betrachtung der Forschungsfrage

### 7.3.1 Erste Randbedingung

**Die Erarbeitung der Updatefähigkeit einer Zielapplikation mit neuer, gültigen Autorität für die Wiederverwendbarkeit des Produktes zur Etablierung einer zeitwertgerechten, ressourceneffizienten Kreislaufwirtschaft**

Die Forschungsfrage ergab sich aus dem Automotive-Aftermarket Kerngeschäft zur Wiederaufbereitung von elektrischen Steuergeräten und Komponenten. Durch das irreversible Setzen einer Verifizierungsinformation (Vertrauensanker, „Root of Trust“) war dieser Prozessor zur Wiederaufbereitung nicht mehr wirtschaftlich, da ganze Prozessoren aufwändig getauscht werden mussten. Hierdurch ergab sich eine signifikante Verschlechterung der Wiederaufbereitungsquote, obgleich die elektrische Fehlerbehebung bereits abgeschlossen war.

Diese Problemstellung weitete sich auch für die Updatefähigkeit eines Gerätes aus. Durch die dynamische Projektentwicklung von neuen Softwarepaketen kam es wiederholt dazu, dass sich auch innerhalb eines Produktionszyklus die Autorität der Applikation geändert hat und somit eine Hardware gleicher Version nicht kompatibel zur neuen Softwareversion wurde.

Durch das Einbringen zusätzlicher Komponenten respektive der dedizierten Autoritäten wurde die Funktionalität eröffnet, die Updatefähigkeit einer Applikation mit einer neuen, gültigen Autorität für die Wiederverwendbarkeit des Produktes zu erreichen.

### 7.3.2 Zweite Randbedingung

**Verbesserung und Erneuerungen von Updatefähigkeiten mit neuer, gültiger Autorität für eine Service-orientierten Bereitstellung neuer Softwarekomponenten**

Mit aufsteigender Domänenübergreifender Kommunikation und lokaler Verarbeitung wird der Ansatz einer zentralisierten Rechnereinheit erforderlich. Service-orientierte Ansätze werden etabliert, die die Architektur mit Eigenschaften wie der Verständlichkeit, dem Systemlebenszyklus, der Erweiterbarkeit, Wartbarkeit und Wiederverwertbarkeit insbesondere im Hinblick auf durchgängige Software- und Systemtechnik beschreiben.

Die dynamische Anpassung der internen Struktur setzt voraus, dass die verschiedenen Indizes im Rahmen eines Softwareupdates und somit auch während der Laufzeit verändert werden müssen und daher die Indizes der jeweiligen Blöcke unabhängig vom Signierungsprozess und außerhalb des

Bereichs liegen, worüber eine Signatur gebildet wird.

## 7.4 Abgrenzung

In der Ausgestaltung der Indizes, welche als vereinfacht dargestellte Verweise dargestellt sind, wurde hier auf eine Einbindung einer konkreten SBOM verzichtet. Die Methodik der Validierungsprozessen wurde auf die Integritäts- und Authentizitätsprüfung beschränkt.

## 7.5 Ausblick

Das Service-orientierte Bereitstellen neuer Softwarekomponenten wird außerhalb des Automotive-Bereiches oftmals direkt über den analog bislang in lokalen Netzwerken praktizierten PXE-Bootvorgang (Preboot Execution Environment) praktiziert.

Der Interne-Struktur-Block kann beispielsweise auch genutzt werden, um eine komplette nachfolgende Instanz aus einem Backend des OEM oder anderer, zertifizierter Parteien zu laden. Wenn eine hinreichend schnelle Internetanbindung verfügbar ist (etwa über 5G-Mobilfunk), kann über das Netzwerkbooten nur eine minimale Funktionalität auf dem Steuergerät selbst vorgehalten werden. Die Applikationen können dann immer jeweils in der aktuellsten Fassung nachgeladen werden.

Der Programmcode kann beispielsweise auch ein trainiertes Machine-Learning-Modul implementieren, wie etwa ein neuronales Netzwerk. Der

Datenblock kann dann beispielsweise Parameter enthalten, die das Verhalten des trainierbaren Moduls charakterisieren. Bei einem neuronalen Netzwerk umfassen diese Parameter z. B. Gewichte, mit denen Eingaben, die einem Neuron oder einer anderen Verarbeitungseinheit zugeführt werden, zu einer Aktivierung dieses Neurons bzw. dieser Verarbeitungseinheit verrechnet werden.

## 7.6 Zusammenfassung

Dieses Kapitel hat die Ergebnisse dieser Arbeit zur Verbesserung der Updatefähigkeit mit neuer, gültiger Autorität dargelegt. Durch das Konzept einer Instanz-orientierten Methodik wurde das Kernziel erreicht, Applikationen und statische Datensätze mit neuer Autorität während der Laufzeit upzudaten. Hierzu stand die Methodik ganz im Automobilkontext. Mechanismen zum primären Schutz der Softwareintegrität wurden eingehalten. Dies wurde durch die Risikoanalyse erreicht. Bekannte Angriffsphänomene im Kontext von basierten Angriffsbäumen wurden in den verschiedenen Iterationen der Modellierung korreliert. Die Ergebnisse dieser Risikoanalyse wurden verallgemeinert und auf eine breite Palette von Systemen anwendbar gemacht.

Die Konzepte der dynamischen Speicherzuordnungsmethodik wurden vorgeschlagen. Dieser letzte Ansatz erleichtert die Nutzung der Kontrollflüsse und Daten innerhalb des Systems.

# Abkürzungen und Symbole

### Abkürzungen

| | |
|---|---|
| **3G** | Dritte Generation des Mobilfunkstandards |
| **5G** | Fünfte Generation des Mobilfunkstandards |
| **A – IVI** | Alliance In-Vehicle Infotainment |
| **AES** | Symmetrisches kryptografisches Verfahren (Advanced Encryption Standard) |
| **AE** | Automotive Electronic |
| **ASIC** | Anwendungsspezifische, integrierte Schaltung |
| **AT** | Austausch-Teil |
| **API** | Application Programming Interface – Programmierschnittstelle |
| **BGA** | Ball Grid Array |
| **CAN** | Controller Area Network, serielles Bussystem |
| **DSRC** | Dedicated Short Range Communication |
| **E/E** | Elektronik/Elektrik |
| **ECALL** | Automatisches Notrufsystem für Kraftfahrzeuge |

| | |
|---|---|
| **ECU** | Electronic Control Unit - Elektrisches Steuergerät |
| **ECID** | Zufallswerte und eindeutige Kennung des Gerätes |
| **EDR** | Event Data Recorder – elektronischer Fahrtenschreiber |
| **FIN** | Fahrzeugidentifizierungsnummer |
| **FMEA** | Fehlermöglichkeiten-Einflussanalyse |
| **FOTA** | Flash over the Air – Softwareupdate über eine Drahtlosschnittstelle |
| **HSM** | Hardware-Security-Modul |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **IETF** | Internet Engineering Task Force |
| **IoT** | Internet of Things – Internet der Dinge |
| **IP – Core** | Intellectual Property Core – vorgefertigter Funktionsblock eines Chipdesigns |
| **JTAG** | Joint Test Action Group – IEEE-Standard 1149.1, Methodik zum Testen und Debuggen integrierter Schaltungen |
| **NIST** | National Institute of Standards and Technology |
| **OBD** | On-Board-Diagnose |
| **OEM** | Original Equipment Manufacturer |
| **OS** | Operating System – Betriebssystem |
| **PCB** | Leiterplatte |

| | |
|---|---|
| **PKI** | Public-Key-Infrastrukturen |
| **PLC** | Product Life Cycle – Produktlebenszyklus |
| **RSA** | Asymmetrisches kryptografisches Verfahren nach Rivest, Shamir und Adleman |
| **SHA** | Secure Hash Algorithm – sicherer Hash-Algorithmus |
| **SRK** | Super Root Key |
| **SoC** | System on Chip – Mikrokontroller |
| **TIER** | Zulieferer |
| **T&RAnalyse** | Threat and Risk – Bedrohungs und Risikoanalyse |
| **TRNG** | True Random Number Generator – hardwarebasierter Zufallszahlengenerator |
| **UUID** | Universal Unique Identifier – eindeutige, benutzerbezogene Seriennummer |
| **CSMS** | Cybersecurity-Managementsystem |
| **SUMS** | Softwareupdate-Managementsystem |

### Symbole und Variablen

| | |
|---|---|
| $\mathbf{f}$ | Funktion |
| $\mathbf{f^{-1}}$ | Inverse-Funktion |
| $\mathbf{R}$ | Zuverlässigkeitsfunktion |
| $\mathbf{I}$ | Integrität |

| | |
|---|---|
| $\mathbf{V}$ | Verifizierungsfunktion |
| $\mathbf{L}$ | Instanz |
| $t$ | Zeit |
| $h$ | Hashwert |
| $i$ | Zähler für eine Runde oder Instanz |
| $c$ | Chiffrat |
| $m$ | Nachricht |
| $k$ | Kryptografischer Schlüssel |
| $r$ | Antwortwert einer in Auftrag gebenenen Berechnnung |
| $ch$ | Challenge – eine anhand einer Rechenvorschrift zu lösende Rechenaufgabe |

# Literaturverzeichnis


[1] S. Spitz, M. Pramateftakis, J. Swoboda, and M. Pramateftakis, *Kryptographie und IT-Sicherheit – Grundlagen und Anwendungen*, 2nd ed. Berlin Heidelberg New York: Springer-Verlag, 2011.

[2] R. B. Multimedia, "Future vehicle ee and software architecture," *GENIVI 2018+*, 2019. [Online]. Available: https://www.genivi.org/future-vehicle-architectures-driving-genivi-work

[3] Baranowski, Kochte, and Wunderlich, "Securing access to reconfigurable scan networks," in *2013 22nd Asian Test Symposium*, Nov 2013, pp. 295–300.

[4] P. Schnarz, "Security patterns for amp-based embedded systems," PhD thesis, Department of Informatics, Clausthal University of Technology, 2018.

[5] F. Semiconductor, "Secure boot on i.mx50, i.mx53, and i.mx 6 series using habv4," online, Oct. 2015, aN4581.

[6] Wirz, "Logistikorientierte ersatzteilvarianten-reduzierung am beispiel der automobilindustrie," PhD thesis, Erlangen-Nürnberg, 2014.

[7] V. Consulting, "Unece csms und sums," *Cybersecurity*, 2021. [Online]. Available: https://consulting.vector.com/de/de/solutions/cybersecurity/unece-csms-and-sums/#c184348

[8] M. Vogt, *Funktionale Sicherheit von Fahrzeugen*. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 307–320. [Online]. Available: https://doi.org/10.1007/978-3-642-30653-2_25

[9] P. Mundhenk, A. Paverd, A. Mrowca, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Security in automotive networks: Lightweight authentication and authorization," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 2, pp. 25:1–25:27, Mar. 2017. [Online]. Available: http://doi.acm.org/10.1145/2960407

[10] M. Bormann, "Konzepterstellung für die optimierung der nachserienfertigung von motorsteuergeräten für die robert bosch gmbh," Master's thesis, HOCHSCHULE WESERBERGLAND, 2018.

[11] Bothe, "Planung und steuerung der ersatzteilversorgung nach ende der serienfertigung," PhD thesis, RWTH Aachen, 2003.

[12] J. Nagel, *Risikoorientiertes Anlaufmanagement*, 1st ed. Berlin Heidelberg New York: Springer-Verlag, 2011.

[13] D. Kiritsis, "Closed-loop plm for intelligent products in the era of the internet of things," *Computer-Aided Design*, vol. 43, no. 5, pp. 479–501, 2011, emerging Industry Needs for Frameworks and Technologies for Exchanging and Sharing Product Lifecycle Knowledge. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448510000485

[14] R. Casper and E. Sundin, "Addressing today's challenges in automotive remanufacturing," *Journal of Remanufacturing*, vol. 8, no. 3, pp. 93–102, Oct 2018. [Online]. Available: https://doi.org/10.1007/s13243-018-0047-9

[15] S. J. Johnston, M. Scott, and S. J. Cox, “Recommendations for securing internet of things devices using commodity hardware,” in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 307–310.

[16] R. Ziolkowski, Ed., *i.MX Applications Processor Trust Achitecture AMF-IND-T0291*, F. Freescale, Sep. 2013.

[17] J. Salecker and D. Schütz, “Bill of material,” in *Proceedings of the 9th European Conference on Pattern Languages of Programms (EuroPLoP ’2004), Irsee, Germany, July 7-11, 2004*, K. Marquardt and D. Schütz, Eds. UVK - Universitaetsverlag Konstanz, 2004, pp. 275–284. [Online]. Available: http://hillside.net/europlop/HillsideEurope/Papers/EuroPLoP2004/2004_SaleckerEtAl_BillOfMaterial.pdf

[18] S. Brunner, J. Roder, M. Kucera, and T. Waas, “Automotive e/e-architecture enhancements by usage of ethernet tsn,” in *2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, June 2017, pp. 9–13.

[19] M. Traub, A. Maier, and K. L. Barbehön, “Future automotive architecture and the impact of it trends,” *IEEE Software*, vol. 34, no. 3, pp. 27–32, May 2017.

[20] S. Cadzow, “Vehicle data standards eco-system,” in *ETSI LI-Rap 53e*. ETSI, Jul. 2020, lI(20)R53002. [Online]. Available: https://docbox.etsi.org/LI/LI/05-CONTRIBUTIONS/2020/LI(20)R53002_Short_overview_of_vehicle_data_standards_eco-system.pdf

[21] J. Buchmann, *Einführung in die Kryptografie*, 6th ed. Berlin Heidelberg New York: Springer-Verlag, 2016.

[22] K. Reif, *Bosch Autoelektrik und Autoelektronik Bordnetze, Sensoren und elektronische Systeme*. Vieweg+Teubner, 2011.

[23] P. Schnabel, *Computertechnik-Fibel – Grundlagen Computertechnik, Mikroprozessortechnik, Halbleiterspeicher, Schnittstellen und Peripherie*, 1st ed. Ludwigsburg: P. Schnabel, 2003.

[24] H. Kim, Y. Won, and S. Kang, "Embedded nand flash file system for mobile multimedia devices," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 545–552, May 2009.

[25] AUTOSAR, *AUTOSAR EXP LayeredSoftwareArchitecture*, 4th ed., Autosar, 2018. [Online]. Available: https://www.autosar.org/fileadmin/Releases_TEMP/Classic_Platform_4.4.0/General.zip

[26] Gematik, "Methode der sicherheitsanalyse in der telematikinfrastruktur," *BSI J*, May 2013. [Online]. Available: https://fachportal.gematik.de/fileadmin/user_upload/fachportal/files/Spezifikationen/Methodische_Festlegungen/gematik_Einheitliche_Methode_Sicherheitsanalyse_V1.1.0.pdf

[27] N. Böcher, "Nachserienversorgungsstrategien für software updates - updatefähigkeit von irreversibel geschützten applikationen," in *Tagungsband Embedded Software Engineering Kongress 2018*, 2018. [Online]. Available: www.ese-kongress.de/downloads

[28] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, Aug 2015, pp. 57–64.

[29] E. A. M. Association, "Acea report - vehicles in use europe," European Automobile Manufactures Association, Tech. Rep., 2021.

[30] B. Z. Online, “Für jedes produkt ein service,” Bilanzpressekonferenz, Interview Volkmar Denner, Apr. 2016, 27. [Online]. Available: https://inside-ws.bosch.com/FIRSTspiritWeb/wcms/wcms_bnn/de/news/news_detail_page_162298.html

[31] N. Böcher, “Effizientes anlaufmanagement für die elektronischen assistenzsysteme - strategien für effiziente softwareupdates für automotive electronics im produktlebenszyklus,” in *13. Fachkonferenz RAMP UP in Leipzig*, 2018.

[32] N. Böcher, “Refurbishment of automotive electronic components regarding update capability of applications,” in *ADAPTIVE 2017-Managed Adaptive Automotive Product Line Development*, 2017. [Online]. Available: https://www.thinkmind.org/articles/adaptive_2017_4_20_58014.pdf

[33] B. für Bildung und Forschung, “Richtlinie zur förderung von forschungs- und entwicklungsvorhaben zum thema "ressourceneffiziente kreislaufwirtschaft – innovative produktkreisläufe,” in *Bundesanzeiger*, 2017. [Online]. Available: https://www.bmbf.de/foerderungen/bekanntmachung-1492.html

[34] D. Stabili, L. Ferretti, and M. Marchetti, “Analyses of secure automotive communication protocols and their impact on vehicles life-cycle,” in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2018, pp. 452–457.

[35] Borgeest and Kai, *Elektronik in der Fahrzeugtechnik - Hardware, Software, Systeme und Projektmanagement*, 2nd ed. Berlin Heidelberg New York: Springer-Verlag, 2010.

[36] E. ITS, “Etsi ts 102 636-4-2 technical specification,” ntelligent Transport Systems (ITS);Vehicular Communications; GeoNetworking; Part

4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 2: Media-dependent functionalities for ITS-G5, techreport 102 636-4-2, Feb. 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/102600_102699/1026360402/01.04.01_60/ts_1026360402v010401p.pdf

[37] T. S. K. Lab, "Mercedes benz mbux security research report," Tencent Security Keen Lab, Tech. Rep., 2021. [Online]. Available: https://keenlab.tencent.com/en/whitepapers/Mercedes_Benz_Security_Research_Report_Final.pdf

[38] M. Hübner and J. Becker, *Multiprocessor System-on-Chip - Hardware Design and Tool Integration*. Berlin Heidelberg: Springer Science & Business Media, 2010.

[39] I. Freescale Semiconductor, *i.MX 6DualPlus/6QuadPlus Applications Processor Reference Manual*, Freescale Semiconductor, Inc., Feb. 2016.

[40] I. Freescale Semiconductor, *Security Reference Manual for i.MX 6Dual, 6Quad, 6Solo, and 6DualLite Families of Applications Processors*, Freescale Semiconductor, 2013.

[41] D. Watjen, *Kryptographie – Grundlagen, Algorithmen, Protokolle*, 3rd ed. Berlin Heidelberg New York: Springer-Verlag, 2018.

[42] A. Boudguiga, W. Klaudel, and J. D. Wesolowski, "On the performance of freescale i.mx6 cryptographic acceleration and assurance module," in *Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, ser. RAPIDO '15. New York, NY, USA: ACM, 2015, pp. 8:1–8:8. [Online]. Available: http://doi.acm.org/10.1145/2693433.2693441

[43] H. Ju, Y. Jeon, and J. Kim, “A study on the hardware-based security solutions for smart devices,” in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec 2015, pp. 833–834.

[44] A. Yamada and T. Ikeda, “Enhanced pki authentication with trusted product at claimant,” *Computers & Security*, vol. 67, pp. 324–334, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404817300044

[45] K. Müller, R. Ulrich, A. Stanitzki, and R. Kokozinski, “Enabling secure boot functionality by using physical unclonable functions,” in *2018 14th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, July 2018, pp. 81–84.

[46] C. W. Fletcher, M. v. Dijk, and S. Devadas, “A secure processor architecture for encrypted computation on untrusted programs,” in *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing*, ser. STC ’12. New York, NY, USA: ACM, 2012, pp. 3–8. [Online]. Available: http://doi.acm.org/10.1145/2382536.2382540

[47] I. Lebedev, K. Hogan, and S. Devadas, “Invited paper: Secure boot and remote attestation in the sanctum processor,” in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, July 2018, pp. 46–60.

[48] S. D. T. Ziola, “Volatile device keys and applications thereof,” 2009.

[49] T. Lu, R. Kenny, and S. Atsatt, “Secure device manager for intel® stratix® 10 devices provides fpga and soc security,” in *The Secure Device Manager for Intel Stratix 10 devices provides a failsafe, strongly authenticated, programmable security scheme for device*

*configuration*. Intel, 2018. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01252-secure-device-manager-for-fpga-soc-security.pdf

[50] Y. Fan, S. Liu, G. Tan, X. Lin, G. Zhao, and J. Bai, "One secure access scheme based on trusted execution environment," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Aug 2018, pp. 16–21.

[51] A. Khan, A. Schaefer, and M. Zetlmeisl, "Efficient memory-protected integration of add-on software subsystems in small embedded automotive applications," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, pp. 44–50, Feb 2007.

[52] K. Kässens, "Ununterscheidbarkeit in der kryptographie," Master's thesis, Leibniz Universität Hannover, 2019. [Online]. Available: https://www.thi.uni-hannover.de/fileadmin/thi/abschlussarbeiten/kaessens-ba.pdf

[53] M. Teuho, E. Pekkarinen, and T. Hämäläinen, "Visualization of memory map information in embedded system design," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 163–166.

[54] Dombrowski and Schulze, *Lebenszyklusorientiertes Ersatzteilmanagement. Neue Herausforderungen durch innovationsstarke Bauteile in langlebigen Primärprodukten*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 439–462. [Online]. Available: https://doi.org/10.1007/978-3-540-75642-2_20

[55] P. Strompen, *Die IATF 16949 - Interpretation der Anforderungen der IATF 16949:2016*. TüV Media GmbH, Köln, 2017.

[56] E. Union, "Schaffung eines rahmens für die genehmigung von kraftfahrzeugen und kraftfahrzeuganhän- gern sowie von systemen, bauteilen und selbstständigen technischen einheiten für diese fahrzeuge," in *RICHTLINIE 2007/46/EG*, 2007. [Online]. Available: https://verkehrslexikon.de/PDF/RiLi_2007-46-EG_Rahmenrichtlinie_Kfz_und_Anhaenger.pdf

[57] U. E. UNITED NATIONS, "Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system," in *UN Regulation No. 155*, Mar. 2021.

[58] U. E. UNITED NATIONS, "Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system," in *UN Regulation No. 156*, Jan. 2021.

[59] A. Nico, *Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262*. Karlsruhe: KIT Scientific Publishing, 2015.

[60] C. H. Li and H. K. Lau, "Integration of industry 4.0 and assessment model for product safety," in *2018 IEEE Symposium on Product Compliance Engineering (ISPCE)*, May 2018, pp. 1–5.

[61] C. H. Li and H. K. Lau, "A critical review of product safety in industry 4.0 applications," in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec 2017, pp. 1661–1665.

[62] D. Howard, "Functional safety and engineering design automation," in *2019 Pan Pacific Microelectronics Symposium (Pan Pacific)*, Feb 2019, pp. 1–9.

[63] E. Hering and A. Schloske, *Zusammenhang zwischen System-FMEA, Konstruktions-FMEA und Prozess-FMEA*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019, pp. 55–58. [Online]. Available: https://doi.org/10.1007/978-3-658-25763-7_6

[64] ISO, "Road vehicles — cybersecurity engineering," in *ISO/SAE 21434:2021*, 2021.

[65] C. Schmittner, Z. Ma, C. Reyes, O. Dillinger, and P. Puschner, "Using sae j3061 for automotive security requirement engineering," in *SAFECOMP Workshops*, 2016.

[66] ISO, "Information technology – security techniques – information security management systems – requirements," in *Norm ISO/IEC 27001*, 2017.

[67] Y. Zhang, P. Shi, C. Dong, Y. Liu, X. Shao, and C. Ma, "Test and evaluation system for automotive cybersecurity," in *2018 IEEE International Conference on Computational Science and Engineering (CSE)*, Oct 2018, pp. 201–207.

[68] F. Garcia, "Automotive cyber security: Lessons learned and research challenges," in *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems*, ser. CS2 '18. New York, NY, USA: ACM, 2018, pp. 19–19. [Online]. Available: http://doi.acm.org/10.1145/3178291.3178295

[69] M. Duncan and P. Roche, "Paving the way towards autonomous driving - tackling soft errors to security challenges," in *2017 IEEE International Reliability Physics Symposium (IRPS)*, April 2017, pp. 2E–1.1–2E–1.8.

[70] S. Gifei and A. Salceanu, "Integrated management system for quality, safety and security in developing autonomous vehicles," in *2017 10th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, March 2017, pp. 673–676.

[71] S. Tonetta, E. Schoitsch, and F. Bitsch, *Computer Safety, Reliability, and Security – SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, September 12, 2017, Proceedings*. Berlin, Heidelberg: Springer, 2017.

[72] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1109–1123. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134001

[73] H. Chen, D. Bao, H. Gao, and J. Cheng, "A security evaluation and certification management database based on iso/iec standards," in *2016 12th International Conference on Computational Intelligence and Security (CIS)*, Dec 2016, pp. 249–253.

[74] M. Staron, *Automotive Software Architectures*. Springer, 2017.

[75] B. Murphy, "Optimizing software development processes," in *2016 IEEE/ACM 4th International Workshop on Conducting Empirical Studies in Industry (CESI)*, May 2016, pp. 4–4.

[76] E. Çelik, S. Eren, E. Çini, and ö. Keleş, "Software test automation and a sample practice for an enterprise business software," in *2017 International Conference on Computer Science and Engineering (UBMK)*, Oct 2017, pp. 141–144.

[77] L. Braun, M. Armbruster, L. Fiege, and E. Sax, “Using a domain model to precisely describe function-agnostic electric/electronic-architectures,” in *AmE 2017 - Automotive meets Electronics; 8th GMM-Symposium*, March 2017, pp. 1–6.

[78] C.-C. Soeldner, *Open Innovation in Embedded Systems*. Berlin, Heidelberg: Springer, 2016.

[79] P. Chlebek, *Praxis der User Interface-Entwicklung – Informationsstrukturen, Designpatterns, Vorgehensmuster*, 1st ed. Berlin Heidelberg New York: Springer-Verlag, 2011.

[80] M. N. Riaz, A. Buriro, and A. Mahboob, “The effect of software development project team structure on the process of knowledge sharing: An empirical study,” in *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Jan 2019, pp. 1–5.

[81] M. Mannion, “Software reuse and reusability based on requirements: Product lines, cases and feature-similarity models,” in *Proceedings of the 20th International Systems and Software Product Line Conference*, ser. SPLC ’16. New York, NY, USA: ACM, 2016, pp. 312–312. [Online]. Available: http://doi.acm.org/10.1145/2934466.2956653

[82] B. Gallina, “Towards enabling reuse in the context of safety-critical product lines,” in *2015 IEEE/ACM 5th International Workshop on Product Line Approaches in Software Engineering*, May 2015, pp. 15–18.

[83] H. ElMaraghy, G. Schuh, W. ElMaraghy, F. Piller, P. Schönsleben, M. Tseng, and A. Bernard, “Product variety management,” *CIRP Annals*, vol. 62, no. 2, pp. 629–652, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007850613001972

[84] B. Singh and S. Gautam, "The impact of software development process on software quality: A review," in *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec 2016, pp. 666–672.

[85] G. Schuh, C. Dölle, J. Kantelberg, and A. Menges, "Identification of agile mechanisms of action as basis for agile product development," *Procedia CIRP*, vol. 70, pp. 19–24, 2018, 28th CIRP Design Conference 2018, 23-25 May 2018, Nantes, France. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212827118300179

[86] Hilbrich, *Platzierung von Softwarekomponenten auf Mehrkernprozessoren - Automatisierte Konstruktion und Analyse für funktionssichere Systeme*. Berlin Heidelberg New York: Springer-Verlag, 2015.

[87] C. Schawel and F. Billing, *Top 100 Management Tools - Das wichtigste Buch eines Managers*, 3rd ed. Berlin Heidelberg New York: Springer-Verlag, 2011.

[88] K. Reif, *Sicherheitsaspekte und funktionale Sicherheit*. Wiesbaden: Vieweg+Teubner Verlag, 2012, pp. 251–282. [Online]. Available: https://doi.org/10.1007/978-3-8348-8658-3_9

[89] Y. Xiong, Q. Zhao, and Y. Zhou, "Manufacturer-remanufacturing vs supplier-remanufacturing in a closed-loop supply chain," *International Journal of Production Economics*, vol. 176, pp. 21–28, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925527316000657

[90] S. Bernard, "Remanufacturing," *Journal of Environmental Economics and Management*, vol. 62, no. 3, pp. 337–351, 2011.

[Online]. Available: http://www.sciencedirect.com/science/article/pii/S0095069611000933

[91] C. Hopman, P. N. Wagner, N. Bergmann, and A. Böttcher, "Development of an innovative repair concept for cfrp parts in cars," *Auto Tech Review*, vol. 5, no. 11, pp. 26–31, Nov 2016. [Online]. Available: https://doi.org/10.1365/s40112-016-1235-3

[92] B. Federal Ministry for the Environment, Nature Conservation and N. S. (BMUB), "Programme for the sustainable use and conservation of natural resources," *German Resource Efficiency Programme II*, 2016.

[93] N. Bey, M. Z. Hauschild, and T. C. McAloone, "Drivers and barriers for implementation of environmental strategies in manufacturing companies," *CIRP Annals*, vol. 62, no. 1, pp. 43 – 46, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007850613000024

[94] C. Laplante, "Improving reliability throughout the product life cycle," in *2018 Annual Reliability and Maintainability Symposium (RAMS)*, 2018, pp. 1–4.

[95] Y. H. Fujita K, "Product variety optimization simultaneously designing module combination and module attributes." 2004.

[96] A. Pc and B. Prabhu, "Integrating requirements engineering and user experience design in product life cycle management," in *Proceedings of the First International Workshop on Usability and Accessibility Focused Requirements Engineering*, ser. UsARE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 12–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=2667081.2667084

[97] T. Belzner, “Leitfaden zur minderung der kostenauswirkungen hoher produktvarianz auf die herstellkosten während der frühen produktentstehungsphasen,” Master’s thesis, wbk Institut für Produktionstechnik Karlsruher Institut für Technologie (KIT), Jun. 2014.

[98] H. Zhang, W. Zhao, J. Zhang, G. Li, and R. Tan, “An approach on optimization, upgrade, renewal of product platform,” in *2006 IEEE International Conference on Management of Innovation and Technology*, vol. 2. IEEE, June 2006, pp. 1108–1112.

[99] A. Hallerbach, T. Bauer, and M. Reichert, “Managing process variants in the process lifecycle,” in *10th Int’l Conf. on Enterprise Information Systems (ICEIS’08)*, June 2008, pp. 154–161. [Online]. Available: http://dbis.eprints.uni-ulm.de/193/

[100] A. Ltd., “Building a secure system using trustzone technology,” ARM Ltd., Tech. Rep., 2009. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[101] R. Perez, “Silicon systems security and building a root of trust,” in *2015 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov 2015, pp. 1–4.

[102] A. Inc., Ed., *iOS-Sicherheit – White Paper*. Apple Inc., Mar. 2017. [Online]. Available: https://www.apple.com/kr/business-docs/iOS_Security_Guide.pdf

[103] V. R. Kebande, N. M. Karie, A. Michael, S. M. Malapane, and H. Venter, “How an iot-enabled “smart refrigerator” can play a clandestine role in perpetuating cyber-crime,” in *2017 IST-Africa Week Conference (IST-Africa)*, 2017, pp. 1–10.

[104] V. der Automobilindustrie e. V. (VDA), “Position zur automotive security,” Verband der Automobilindustrie e. V. (VDA), Tech. Rep., 2017.

[105] B. Deutschland, “Produkthaftungsgesetz,” *Gesetz vom 17.07.2017 (BGBl. I S. 2421) m. W. v. 22.07.2017*, 1989. [Online]. Available: https://www.gesetze-im-internet.de/prodhaftg/

[106] B. fur Sicherheit in der Informationstechnik, “Kryptographische verfahren: Empfehlungen und schlussellangen,” Online, May 2018. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile

[107] A. Chaouch, B. Bouallegue, and O. Bouraoui, “Software application for simulation-based aes, rsa and elliptic-curve algorithms,” in *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, March 2016, pp. 77–82.

[108] P. Zhang and J. Fang, “Comparing and implementation of public key cryptography algorithms on smart card,” in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 12. ICCASM, Oct 2010, pp. V12–508–V12–510.

[109] A. Ltd., “Trustzone arm developer,” 2018. [Online]. Available: https://developer.arm.com/technologies/trustzone

[110] N. Ayres, L. Deka, and B. Passow, “Virtualisation as a means for dynamic software update within the automotive e/e architecture,” in *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud*

*Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019, pp. 154–157.

[111] S. Kugele, P. Obergfell, M. Broy, O. Creighton, M. Traub, and W. Hopfensitz, "On service-orientation for automotive software," in *2017 IEEE International Conference on Software Architecture (ICSA)*, April 2017, pp. 193–202.

[112] P. Schnarz, J. Wietzke, and I. Stengel, "Towards attacks on restricted memory areas through co-processors in embedded multi-os environments via malicious firmware injection," in *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*, ser. CS2 '14. New York, NY, USA: ACM, 2014, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/2556315.2556318

[113] D. E. ISO, *DIN EN ISO/IEC 18045- Informationstechnik - Sicherheitstechniken - Methodik Für Die Bewertung Der IT-Sicherheit (ISO/IEC 18045:2008) - Information Technology - Security Techniques - Methodology for IT Security Evaluation (ISO/IEC 18045:2008)*. Frankfurt am Main: Beuth Verlag GmbH, 2008.

[114] L. Schnieder and R. S. Hosse, *Leitfaden Automotive Cybersecurity Engineering - Absicherung vernetzter Fahrzeuge auf dem Weg zum autonomen Fahren*. Berlin Heidelberg New York: Springer-Verlag, 2018.

[115] I. S. Organisation, "Information technology – security techniques – methodology for it security evaluation," *ISO/IEC 18045:2005*, 2014.

[116] N. Böcher, A. Rausch, and A. Wenda, "Steuergerät sowie verfahren zu dessen betrieb," in *Patent, Aktenzeichen DE: 102018211139.1*. Deutsches Patent- und Markenamt, 2018.

[117] N. Böcher and G. Blott, “Steuergerät und verfahren zum betreiben desselben,” in *Patent, Aktenzeichen DE: 102019210625.0*. Deutsches Patent- und Markenamt, 2019.

[118] N. Böcher and G. Blott, “Verfahren zum betreiben einer aktualisierbaren applikation auf einem steuergerät,” in *Patent, Aktenzeichen DE 102019216141.3*. Deutsches Patent- und Markenamt, 2019.

[119] N. Böcher and G. Blott, “Verfahren zum betreiben eines steuergeräts, auf dem mehrere applikationen ausgeführt werden,” in *Patent, Aktenzeichen DE: 102020216380.4*. Deutsches Patent- und Markenamt, 2020.

[120] N. Böcher and G. Blott, “Verfahren zum betreiben eines steuergeräts, auf dem mehrere applikationen ausgeführt werden,” in *Patent WO 2022/136083 A1*. World Intellectual Property Organization (WIPO), 2022.

[121] N. Böcher and G. Blott, “Method for operating control device execuring plurality of applications,” in *Patent CN 116635854*. CN, 2023.

[122] N. Böcher and G. Blott, “Verfahren zum betreiben eines steuergeräts, auf dem mehrere applikationen ausgeführt werden,” in *Patent EP 4264421 A1*. EP, 2023.

[123] N. Böcher and G. Blott, “Method for operating a control unit on which multiple applications are executed,” in *Patent US 2023/0385076 A1*. U.S. Patent and Trademark Office, 2023.

[124] N. Böcher and G. Blott, “Method for operating a control unit on which multiple applications are executed,” in *Patent JP 2024500453 A*. JP, 2024.

[125] H. Siregar, E. Junaeti, and T. Hayatno, “Implementation of digital signature using aes and rsa algorithms as a security in disposition system af letter,” *IOP Conference Series: Materials Science and Engineering*, vol. 180, p. 012055, mar 2017. [Online]. Available: https://doi.org/10.1088%2F1757-899x%2F180%2F1%2F012055

[126] Toradmalle, Singh, Shastri, Naik, and Panchidi, “Prominence of ecdsa over rsa digital signature algorithm,” in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on*, Aug 2018, pp. 253–257.

[127] K. K., C. Rebeiro, and A. Hazra, “An algorithmic approach to formally verify an ecc library,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 5, pp. 63:1–63:26, Aug. 2018. [Online]. Available: http://doi.acm.org/10.1145/3224205

[128] L. et. al., *Public Key Infrastructure – 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, June 28-30, 2007, Proceedings*. Berlin Heidelberg: Springer Science & Business Media, 2007.

# A Anhang

## A.1 Kryptografische Prüfsumme am Beispiel SHA-1 und SHA-2

Bei umfangreichen Dokumenten oder verwendeter Software kann der Zeitbedarf sehr groß sein, um diese mit einem Integritätsschutz, wie beispielsweise einer Verschlüsselung, versehen zu können. Ein Lösungsansatz ist der Einsatz einer schnell zu berechnenden kryptografischen Prüfsumme (Hashwert). Abbildung A.1 beschreibt die Bildung eines Hashwertes h(m) mittels einer kryptografischen Einwegfunktion h, die aus einer Nachricht m von beliebiger Länge erzeugt wird.

Im Kern wird anhand einer großen Nachricht m eine kleinere Zeichenfolge h(m) erstellt, die über die Eigenschaft einer Kollisionssicherheit verfügt. Nach Swoboda et al. [1] soll eine gute Hashfunktion kollisionsresistent sein. So sollte ein Hashwert h(m) nicht gleichwertig mit einem Hashwert einer Nachricht h(m+1) sein. Auch ist über die Einwegfunktion sichergestellt, dass ausgehend eines Hashwertes h(m) nicht der originale Inhalt der Nachricht m erzeugt werden kann.

Auch spielen Hashwerte eine wichtige Rolle in Datenbanksystemen. Über jedes Objekt der Datenbank wird ein Hashwert berechnet. Abgelegt in

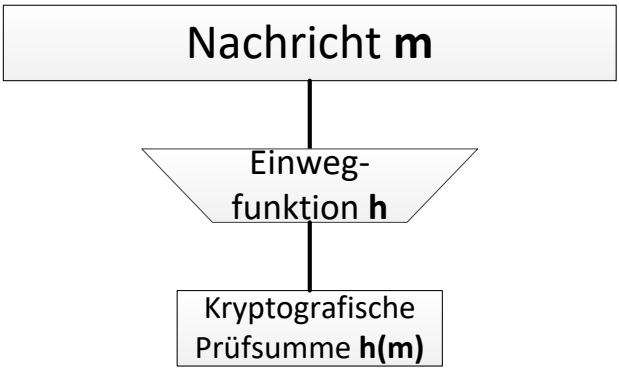


**Abbildung A.1:** Einweg-Hashfunktion

einer kompakten Hashtabelle, kann der aus einer Suchanfrage berechnete Hashwert leicht mit deren Einträgen verglichen werden.

Wie im weiteren Verlauf in Abschnitt A.2.2.2 beschrieben, kann dieser Hashwert verschlüsselt und als Signatur verwendet werden.

Im folgenden Beispiel wird auf den Secure-Hash-Algorithmus SHA eingegangen, der von dem National Institute of Standards and Technology (NIST) veröffentlicht wurde. Als Grundlage soll zunächst der bereits abgelöste Vorgänger SHA-1 und anschließend der aktuelle Standard der SHA-2-Familie erläutert werden.

### A.1.1 SHA-1

Der SHA-1-Algorithmus arbeitet rundenbasiert und generiert aus einer beliebig langen Nachricht m einen Hashwert h(m) mit der Länge von 160 Bit. Die Nachricht m wird in Blöcken von 512 Bit Länge geteilt. Der letzte Block wird durch Auffüllung (padding) und eine Längenangabe gebildet. Somit ergeben sich pro Block einer Nachricht m 16 Wörter $W_i (i = 0..15)$ zu je 32 Bit. Jeder Block wird in 80 Runden verarbeitet. Zur Verarbeitung eines Blockes in 80 Runden werden nun die 16 Wörter eines Blockes auf 80 Wörter expandiert. Diese Expansion ist in Gleichung A.1 dargestellt und zeigt die Bildung der restlichen Wörter $W_i$ mit $i = 16$ bis $i = 79$. $\oplus$ stellt eine Addition mod 2 (XOR) dar. Ein zyklische Links-Rotation von einer Stelle ist mit „$<<< 1$“ dargestellt.

$$W_i = <<< 1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \;\; für\; i = 16...79 \tag{A.1}$$

Für die finale Berechnung der 160 Bit des Hashwerts h(m) werden diese 160 Bit in 5 Worte à 32 Bit unterteilt, die im Folgenden mit A, B, C, D und E bezeichnet werden und über einen definierten Anfangswert verfügen[1]. Abbildung A.2 zeigt einen Rundendurchlauf, der insgesamt 80 Mal pro Block der Nachricht m vollzogen wird.

F ist eine Funktion, die einen 32-Bit-Wert ergibt, „+“ eine Addition Modulo $2^{32}$. $K_i$ ist eine 32-Bit-Konstante für die Runde i und $W_i$ das jeweilige 32-Bit-Wort der Nachricht m. Die Funktion F ist abhängig von der Runde i.

[1] Definition der fünf 32-Bit-Anfangswerte: A = 0x67452301, B = 0xefcdab89, C = 0x98badcfe, D = 0x10325476, E = 0xc3d2e1f0. [41]

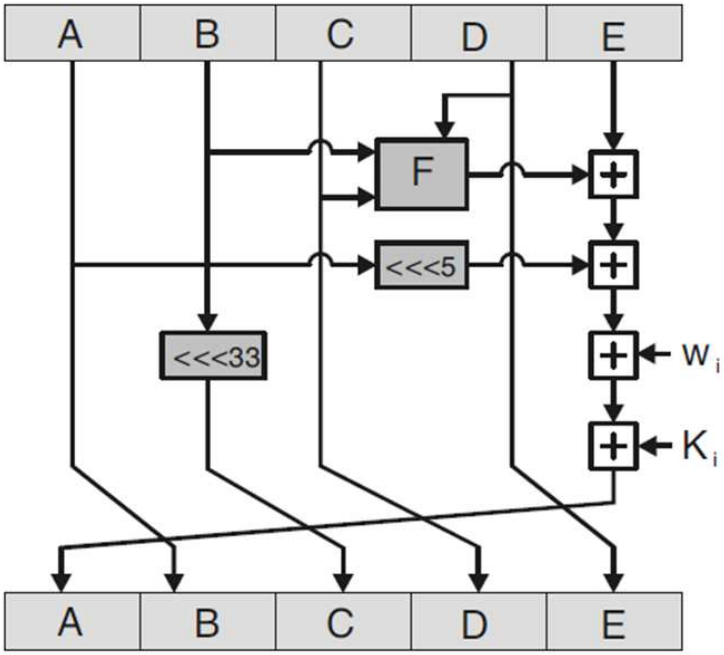


**Abbildung A.2:** SHA-1-Algorithmus nach [1]

Nach jeweils 20 Runden wechselt die Funktionalität, die in Gleichung A.2 beschrieben ist. Die 32-Bit-Konstante $K_i$ ist ebenfalls für jeweils 20 Runden gleich. Die entsprechenden 4 Konstanten für die 4*20 = 80 Runden

sind fest definiert[2].

$$\begin{aligned} F_i(B,C,D) \;=\; (B \wedge C) \vee (\neg B \wedge D) \;\; für\, i = \; 1...19 \\ B \oplus C \oplus D \;\; für\, i = 20...39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \;\; für\, i = 40...59 \\ B \oplus C \oplus D \;\; für\, i = 60...79 \end{aligned} \tag{A.2}$$

### A.1.2 SHA-2

Unter dem Begriff der SHA-2-Familie versteht man die Hashfunktionen SHA-256/224 und SHA-512/384. Der signifikante Unterschied liegt in der Anzahl der Rundendurchläufe und der verwendeten Wörter eines Blockes der Nachricht m. somit die Anzahl der resultierenden Wörter (A, B, . . . ) für den resultierenden Hashwert h(m).

Beispielsweise funktioniert SHA-256 analog zu der Unterteilung der SHA-1-Blöcke von 512 Bit der Nachricht m, die in 8 Wörter zu je 32 Bit unterteilt sind. Zudem arbeitet der SHA-2-Algorithmus mit i = 64 Runden. Somit werden auch die 8 Wörter eines Blockes zu je 64 Wörtern mit $W_i(i = 0..63)$ expandiert. Für die finale Berechnung des Hashwerts h(m) ergeben sich 256 Bit mit 8 resultierenden Wörtern (A, ..., H). Die Version SHA-224 wird bei gleicher Berechnung im Endergebnis auf 224 Bit reduziert.

Beim SHA-512-Algorithmus ist die Nachricht m in 1024-Bit-Blöcke aufgeteilt, bestehend aus 16 Wörtern zu je 64 Bit, ebenfalls expandiert auf 80 Wörter, jedoch zu je 64 Bit. Für die finale Berechnung des Hashwerts

[2] Konstante $K_i$ mit 0x5a827999, 0x6ed9eba1, 0x8f1bbcdc, 0xca62c1d6. [41]

h(m) ergeben sich 512 Bit mit 8 resultierenden Wörtern (A, ..., H) bei einer Verarbeitungsbreite von 64 Bit. Die Version SHA-384 wird bei gleicher Berechnung im Endergebnis auf 384 Bit reduziert. [1]

## A.2 Kryptografische Grundlagen symmetrischer und asymmetrischer Verschlüsselungen

Nach Wätjen [41] kam die Kryptografie schon bei alten Kulturvölkern wie den Ägyptern zum Einsatz und wird als Lehre vom geheimen Schreiben bezeichnet. Die Methodik der Verschlüsselung einer Botschaft fand in vielen Bereichen Anwendung, so z. B., um Schriften mit ohnehin wenigen Lesern wichtiger erscheinen zu lassen, jedoch auch im geschäftlichen Interesse. Vor allem wurde die Kryptografie schon früh für nachrichtentechnische Zwecke verwendet – ob beim Militär oder in der Diplomatie. Im Kern ging es um die Ver- und Entschlüsselung durch einen Sender bzw. Empfänger, allerdings auch um den Versuch des Brechens abgefangener Geheimtexte. Diese beiden Prozesse beeinflussten sich und wurden stetig weiterentwickelt. Ihre Anwendung wurde von Spezialisten vollzogen, im alltäglichen Leben fanden sie keinen Gebrauch. So wurden lange Zeit im herkömmlichen Briefverkehr Briefe einfach in einem Umschlag verschickt, ggf. zugeklebt oder versiegelt. Beim Empfänger wurden diese dann weggeschlossen. Diese traditionellen Sicherheitsmaßnahmen kommen aufgrund der digitalen Transformation jedoch nicht mehr zum Einsatz. Bei dem elektronischen Transport von E-Mails und Messenger-Nachrichten ist es notwendig, sicherzustellen, dass die Nachricht auf dem Transportweg nicht einfach ausgelesen werden kann, sondern verschlüsselt ist.

In Anlehnung an das Schichtenmodell ISO/IEC 7498 ist nicht nur die Sicherung der Transportschicht erforderlich, sondern auch die Sicherung der Daten und deren Verwahrung selbst. Praktisch gesehen kann ein Datenträger unverschlüsselte Texte beinhalten und diese Drittparteien zur Verfügung stellen. Somit muss nach Buchmann [21] auch ihre Fälschung ausgeschlossen sein.

In den folgenden Abschnitten dieses Unterkapitels werden daher Grundlagen der symmetrischen Verschlüsselungen unter A.2.1 am Beispiel des AES-Verfahrens und der (asymmetrischen) Public-Key-Kryptosysteme unter A.2.2 mit Hilfe des RSA-Verfahrens erläutert. Zudem wird explizit anhand der in Abschnitt A.1 erklärten Hashfunktionen auf Signierungsprozesse unter A.2.2.2 eingegangen.

### A.2.1 Symmetrische Verschlüsselung

Im folgenden Abschnitt werden Grundlagen über die symmetrische Verschlüsselung vorgestellt. Im Kern ist ein geheimer Schlüssel $k$ in Verwendung, um ein Chiffrat $c$ einer Nachricht $m$ zu erstellen. Dieser Schlüssel $k$ wird auch zur Entschlüsselung verwendet. Gleichung A.3 und Gleichung A.4 beschreiben die Funktion der Ver- und Entschlüsselung aus den Beschreibungen kryptografischer Grundlagen nach [21], [41], [1] und [52].

$$Verschlüsselung \; : \; c \; = \; f(k, m) \; = \; f_k(m) \tag{A.3}$$

$$Entschlüsselung \; : \; m \; = \; f^{-1}(k, c) \; = \; f_k^{-1}(c) \tag{A.4}$$

Die Ver- und Entschlüsselungsfunktion lassen sich darüber hinaus in zwei Charakteristika unterteilen.

#### A.2.1.1 Block-orientierte Verschlüsselung

Eine Block-orientierte Verschlüsselung teilt die zu verschlüsselnde Nachricht $m$ in Blöcke auf, die durch einen Schlüssel $k$ über eine Funktion $f$ jeweils zu einem Chriffrat c gebildet werden. Abbildung A.3 zeigt ein solches Beispiel, indem eine Nachricht $m$ in 4 Blöcke $m_1$, $m_2$, $m_3$ und $m_4$ aufgeteilt ist, die zu einem jeweiligen Chifrat $c_1$, $c_2$, $c_3$ und $c_4$ berechnet werden. Die Länge eines Blockes ist in gängigen Verfahren auf eine feste Bit-Länge definiert. Beispielsweise liegt diese Länge im nachfolgend erläuterten AES-Verfahren im Abschnitt A.2.1.3 bei 128 Bit.

Die vorgestellte Anwendung einer Block-orientierten Verschlüsselung mit gleicher Funktion ist jedoch anfällig für Kryptoanalysen. Gleiche Klartextblöcke ergeben beim gleichen Schlüssel reproduzierbare verschlüsselte Blöcke. Somit lassen sich Blöcke mit Schlüsselwörtern wie Leerzeichen oder monoton ausgefüllte Blöcke identifizieren. Angriffe mit bekanntem Klartext können so leicht verwendet werden.

#### A.2.1.2 Strom-orientierte Verschlüsselung

Bei einer Strom-orientierten Verschlüsselung nach Abbildung A.4 sind Chiffraten $c_n$ durch eine ganze Folge von Zeichen-orientierten Ketten einer Nachricht $m_n$ mit einem Strom von Schlüssel-Bit $k_n$ beziehungsweise

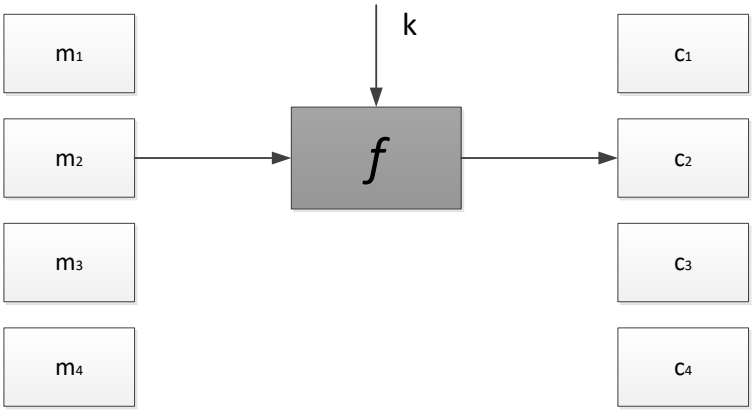


**Abbildung A.3:** Block-orientierte Verschlüsselung

ebenfalls Zeichenketten über eine Rechenoperation $f$ verschlüsselt. Gängige Rechenoperationen sind üblicherweise die Addition Modulo 2 oder Modulo 26.

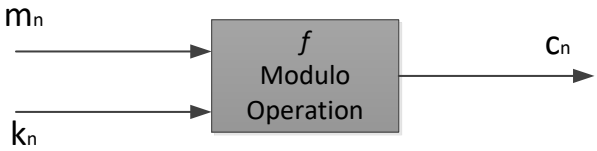


**Abbildung A.4:** Strom-orientierte Verschlüsselung

Mit der Annahme einer echten Zufallsfolge als Schlüsselfolge nach [41] würde eine perfekte Sicherheit erreicht werden. Bei gängigen Verfahren wiederholt sich eine Schlüsselfolge jedoch periodisch. Weitere Möglichkeiten bestehen aus Pseudozufallsfolgen. Diese Art der Verschlüsselung wird vorzugsweise bei Funkübertragungen und im Mobilfunksystem GSM eingesetzt. [1]

#### A.2.1.3 Beispiel: Advanced Encryption Standard AES

Bei dem Advanced Encryption Standard AES handelt es sich wie in der vorangegangenen Grundlage um eine symmetrische Block-orientierte Verschlüsselung. Zum Ver- und Entschlüsseln ist in dieser Standardisierung derselbe Schlüssel $k$ mit einer Länge von 128-, 192- oder 256 Bit in Verwendung. Die zugehörige Blocklänge beträgt 128 Bit. Folgende Erläuterung nach [21], [41], [1] und [125] stellt eine vereinfachte Beschreibung bereit:

Das AES-Schema wird nach [21] als Substitutions-Permutations-Netzwerk bezeichnet. Eine Nachricht m wird in Daten-Blöcken $d$ von 128 Bit Länge geteilt. Abbildung A.5 zeigt einen idealen Durchlauf einer Verschlüsselungsrunde.

Am Anfang steht der zu verschlüsselnde Daten-Block $d$. Dieser 128 Bit lange Block wird einer Transformationsoperation unterzogen, die sich in mehreren Rechenoperationen aufteilt. Beginnend mit einer Substitution werden die Datenblöcke permutiert und vermischt. Parallel wird eine Expansion des AES-Schlüssels vollzogen, um den jeweiligen transformierten Daten-block mit einem Runden-spezifischen Teilschlüssel zu addieren

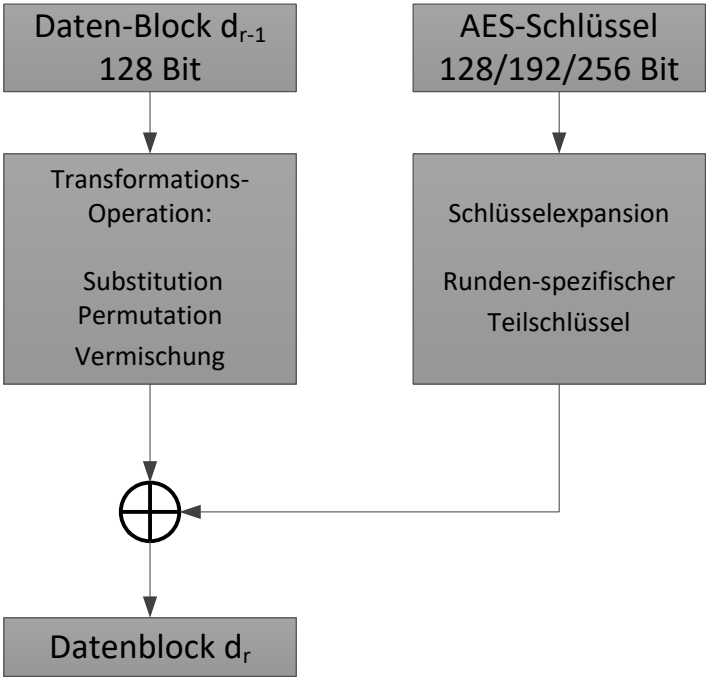


**Abbildung A.5:** Rundendurchlauf nach [21]

(Modulo-2-Addition). Die vollzogenen Rundendurchläufe $r$ zur Verschlüsselung eines Daten-Blockes $d_r$ hängen von der AES-Schlüssellänge ab, beispielsweise 14 Runden bei einer Schlüssellänge von 256 Bit. Für die erste Runde mit $r = 0$ findet einmalig eine Vorrunde statt. Hier entfallen die Transformationsoperationen für den Datenblock $d$. In der letzten Runde entfällt innerhalb der Transformationsoperation des Datenblockes $d$ die

Vermischung.

### A.2.2 Asymmetrische Verschlüsselung

Neben den genannten Grundlagen zur symmetrischen Verschlüsselung in A.2.1 gibt es zudem asymmetrische kryptografische Verfahren. Der Kernansatz bei asymmetrischen Verfahren ist, dass hier ein Schlüsselpaar verwendet wird, bestehend aus einem geheimen privaten und einem öffentlich bekannten Schlüssel. Durch diese Unterteilung muss jedoch sichergestellt sein, dass die Schlüssel nicht manipuliert werden können. Dies trifft insbesondere auf den Integritätsschutz des öffentlichen Schlüssels zu.

Abbildung A.6 zeigt die Grundlage eines asymmetrischen kryptografischen Verfahrens. Die Verschlüsselung einer Klartextnachricht $m$ erfolgt über einen Algorithmus $f$ mit dem öffentlichen Schlüssel $e$ zum Chiffretext $c$. Zum Entschlüsseln wird in einer sicheren Umgebung der Chiffretext $c$ mit dem privaten Schlüssel $d$ über einen Algorithmus $f$ entschlüsselt. Es entsteht die Klartextnachricht $m$. Das mit bekannteste Verfahren ist das nach Rivest-Shamir-Adleman (RSA), das im Folgenden unter A.2.2.1 beschrieben ist.

Asymmetrische Schlüssel bieten gleich mehrere Möglichkeiten. Als zusätzlicher Aspekt kann auch eine Public Key Infrastructure (PKI) angewendet werden, um weitgehende Berechtigungskonzepte eines Systems zu managen. Hierauf wird im Abschnitt A.3 vertieft eingegangen. Der wesentliche Aspekt ist jedoch die Bekanntmachung des öffentlichen Schlüssels in einem System, das in Abbildung A.7 dargestellt ist. Bei einer Kommunikation mehrerer Teilnehmer muss somit nicht jede Konstellation einzeln mit

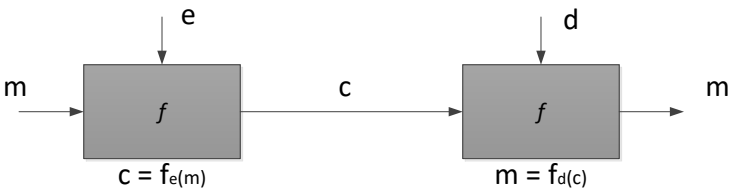


**Abbildung A.6:** Asymmetrisches kryptografisches Verfahren

einem Key versehen werden, sondern nur der zum Teilnehmer gehörige öffentliche Schlüssel.

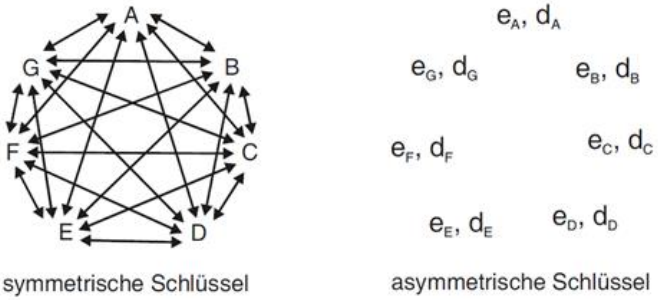


**Abbildung A.7:** Menge der Schlüssel im symmetrischen und asymmetrischen Verfahren nach [21]

Eine weit verbreitete Einsatzmöglichkeit ist die Erstellung einer Signatur unter Verwendung von Hashwerten (siehe Abschnitt A.1), die im nachfolgenden Abschnitt unter A.2.2.2 erläutert ist. Im Kern wird durch den geheimen, privaten Schlüssel eine Signatur erstellt, die von dem öffentlichen Schlüssel innerhalb eines Verifizierungsprozesses validiert werden kann. Die Anwendung findet sich insbesondere im Secure-Boot-Prozess wieder, der im Abschnitt 2.2 und insbesondere im weiteren Verlauf dieser Arbeit unter 5.1.1 behandelt wird.

#### A.2.2.1 Beispiel: Rivest-Shamir-Adleman-Verfahren (RSA)

Das Rivest-Shamir-Adleman-Verfahren (RSA) ist das am häufigsten verwendete Vorgehen in Public-Key-Kryptosystemen. Die Schlüsselerzeugung eines Key-Paares lässt sich nach [21], [41], [1] und [126] in 5 Schritten generieren, die als Formel in Gleichung A.5 bis Gleichung A.9 dargestellt sind:

$$1.\ Generierung\ zweier\ groer\ Primzahlen\ p\ und\ q \tag{A.5}$$

$$\begin{aligned} 2.\ Es\ gilt\ der\ Faktor\ n\ mit\ n = p * q \\ sowie\ \varphi(n) = (p-1)(q-1) \end{aligned} \tag{A.6}$$

$$3. Die\ Zahl\ e \in \mathbb{N} \Rightarrow \quad 1 < e < \varphi(n), mit\ ggT(e, \varphi(n)) = 1 \tag{A.7}$$

$$4. Der\ private\ Schlüssel\ ergibt\ sich\ aus: d = e^{-1} mod \varphi(n) \quad \text{(A.8)}$$

$$5.\ Der\ öffentliche\ Schlüssel\ ergibt\ sich\ aus\ n\ und\ e \quad \text{(A.9)}$$

Für die Verschlüsselung wird der öffentliche Schlüssel $d$, bestehend aus $n$ und $e$, verwendet. Hier wird die Nachricht $m$ als Zahl dargestellt und das Chiffrat $c$ mit der Berechnung aus Gleichung A.10 verschlüsselt:

$$c \ = \ m^e mod\ n \quad \text{(A.10)}$$

Die Entschlüsselung erfolgt mit dem privaten Schlüssel $d$ der Berechnung aus Gleichung A.11:

$$m \ = c^d mod\ n \quad \text{(A.11)}$$

#### A.2.2.2 Signierungsprozess am Beispiel des RSA

Mit der Grundlage einer Blockchiffre am Beispiel des RSA-Verfahrens sowie der Berechnung eines Hashwertes aus Abschnitt A.1 wird im folgenden Abschnitt der Signierungsprozess einer Nachricht $m$ am Beispiel des RSA-Verfahrens dargestellt. Gleichung A.12 beschreibt das RSA-Public-Key-Signaturverfahren. Die Signatur $s$ ergibt sich aus der Nachricht $m$ mit dem Exponenten des privaten Schlüssels $d$, Modulo des Faktors $n$.

$$s \ = \ m^d \ mod\ n \quad \text{(A.12)}$$

Die Signierung bedarf eines erhöhten Rechenaufwands. Statt der Signatur einer langen Nachricht $m$ zu erstellen, wird daher nur der ermittelte Hashwert h(m) signiert.

Die Verifizierung der Signatur erfolgt durch den öffentlichen Schlüssel $e$ und ist in Gleichung A.13 beschrieben. Voraussetzung hierfür ist, dass der öffentliche Schlüssel $e$ als Schlüsselpaar zum Besitzer des privaten Schlüssels $d$ gehört. Um diese Sicherheit der Zuordnung zwischen dem öffentlichen Schlüssel und dem Besitzer zu garantieren, werden Zertifikate verwendet. Ein Zertifikat wird durch eine Zertifizierungsinstanz mittels einer eigenen Signatur beglaubigt.

$$m = s^e \, mod \, n \tag{A.13}$$

Soll zusätzlich die gesamte Nachricht $m$ nach dem RSA-Verfahren verschlüsselt werden, bedeutet dies neben dem erwähnten erhöhten Rechenaufwand auch einen erheblichen zeitlichen Nachteil (um Faktor 1000) zum symmetrischen Verschlüsselungsverfahren, z.B. nach AES. Deshalb wird hier ein Sitzungsschlüssel $k$ generiert und die Nachricht $m$ symmetrisch verschlüsselt. Der Sitzungsschlüssel $k$ wird dann mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. [125]

Die Anwendungen von RSA sind somit die Erstellung einer Signatur auf den Hashwert einer Nachricht h(m) und die Übertragung eines Sitzungsschlüssels $k$ für die hybride Kryptografie.

### A.2.3 Kryptosysteme mit elliptischen Kurven

Die Verwendung algebraischer Konstrukte zu kryptografischen Zwecken entstand Mitte der 80er Jahre unabhängig von Miller und Koblitz. Die darauf basierenden Methoden werden als ECC (Elliptic Curve Cryptography) bezeichnet. Nach [1] und [126] können mit Hilfe elliptischer Kurven effiziente asymmetrische Kryptosysteme entworfen werden. Aufgrund der Eigenschaft mathematischer Gruppierungen kann eine Ellipse aus bestehenden Punkten innerhalb eines kartesischen Koordinatensystems definiert werden. Hierbei werden Punkte festgelegt, an denen die Linie die Abszisse und Ordinate schneidet. Werden Schnittpunkte mit einer Zahl multipliziert, so erhält man einen anderen Punkt auf der Kurve. Folglich ist es sehr schwierig, herauszufinden, welche Zahl hierbei gewählt wurde, selbst wenn das Ergebnis bekannt ist.

Der wesentliche Vorteil liegt in der daraus resultierenden großen kryptografischen Stärke. So kann im Vergleich zum RSA-Verfahren ein Schlüssel mit 1024 Bit Länge bei einem auf elliptischen Kurven basierenden System einen 160 Bit langen Schlüssel aufweisen. [127]

## A.3 Public-Key-Infrastrukturen (PKI)

Der Bedarf am Handling der Schlüsselverwaltung bei asymmetrischen Kryptoverfahren zeichnet sich dadurch aus, dass der öffentliche Schlüssel im Sinne der Datenintegrität abgesichert, aber nicht geheim gehalten werden muss. Die Verteilung und Speicherung der öffentlichen und privaten Schlüsselpaare werden in Public-Key-Infrastrukturen (PKI) gehandhabt. Dieser Abschnitt beschreibt allgemeine Organisationsprinzipien einer PKI

und geht im weiteren Verlauf auf standardisierte PKI-Verfahren, Anwendungen und Formatierungen aus [42], [43] und [44] ein.

### A.3.1 PKI-Management

Die Art und Dauer der Aufbewahrung eines privaten oder öffentlichen Schlüssels hängen ganz von ihrer Verwendung ab. Prinzipiell muss nach Lopez et al. [128] ein öffentlicher Schlüssel, beispielsweise für eine Signaturüberprüfung, solange aufbewahrt werden, dass entsprechende Signaturen weiterhin verifizierbar sind. Der öffentliche Schlüssel wird als digitales Zertifikat gespeichert, das ein Eigentümer sowie seinen Eigenschaften (Gültigkeitsdauer, Berechtigungsstufen, etc.) authentifiziert. Der private Schlüssel muss ebenfalls solange aufbewahrt werden, bis die neue Signierung einer Software angelegt wurde. Die Archivierung der privaten und öffentlichen Schlüssel ist jedoch verschieden.

Die größte Herausforderung liegt in der Verwaltung der Schlüssel über den gesamten Lebenszyklus. Hier steht im Kern das Berechtigungskonzept im Fokus. Nur autorisierte Benutzer, etwa die eines Projektteams oder von zentralisierter Stelle, haben die Befugnis, ein Schlüsselpaar zu erstellen und dessen Verwendung an andere Benutzer zu delegieren. Auch die Schlüsselerstellung selbst muss einer kontrollierten und sicheren Basis unterliegen. So muss gewährleistet sein, dass ein generiertes Schlüsselpaar eindeutig ist und einer Zuordnung entspricht. Auch sind genaue Protokollierungen vorzunehmen, die beispielsweise Zeitstempel der Erstellung, Verwendung oder Löschung beinhalten. Außerdem müssen die Freigabeberechtigung eines privaten Schlüssels und deren Kontrolle rückverfolgbar sein.

Auch auf Produktebene ist ein entsprechendes PKI-Management notwendig. Hier spielt im Wesentlichen der Schutz vor Manipulation (Integritätsschutz), aber auch die Autorisierung eine Rolle. Im Rahmen des technischen Aufbaus eines Embedded Systems wird im Folgenden auf Hardware-Security-Module unter 2.2.2 eingegangen.

### A.3.2 PKI-Standardisierungen

Die Internet Engineering Task Force IETF hat bei PKI-Technologien ebenfalls digitale Signaturen berücksichtigt. Die zugehörige Arbeitsgruppe (PKIX) erstellte die X.509-Spezifikation „Internet X.509 Public Key Infrastructure Certificate and CRL Profile“. Diese beinhaltet die Beschreibung des Formates für Zertifikate und Widerrufslisten anhand der Abstact Syntax Notation ASN1, einer Beschreibungssprache für Datenformate, aber auch Standardisierungen anderer Arbeitsgebiete, wie Zeitstempeldienste und Attributszertifikate als Grundlage für eine weitere Verbreitung von PKIs. So sind, ausgehend von der Firma RSA Laboratories, mit anderen Unternehmen die PKCS-Standards (Public Key Cryptographic Standards) verabschiedet worden. Im Folgenden werden die wichtigsten nach [1] und [42], [43] aufgelistet:

- PKCS # 1: Standard zum RSA-Algorithmus, Definition aller Prozesse zur Signaturerstellung und Verschlüsselung
- PKCS # 3: Beschreibung des Diffie-Hellmann-Protokolls für den sicheren Schlüsselaustausch
- PKCS # 7: Cryptographic Message Syntax

- PKCS # 10: Ein Format für einen Zertifikatsantrag an eine Zertifikatsautorität (CA)
- PKCS # 11: Hier wird eine Schnittstelle (API) in C-Syntax für die Verwendung von Basisdiensten beschrieben. Im Kern kann diese API in einer Software oder einem speziellen HSM (Hardware-Security-Module) verwendet werden. Ziel ist es, eine allgemeine Schnittstelle zu schaffen und diese zur Integration verschiedener Kryptokomponenten (Token) in sicherheitskritische Applikationen zu implementieren.
- PKCS # 12: „Personal Information Exchange Syntax“ spezifiziert ein Format, um Zertifikate und zugehörige private Schlüssel in einer passwortgeschützten Datei abzulegen.
- PKCS # 15: Ein im Zusammenhang mit Chipkarten genutztes Format, um Schlüssel und Zertifikate auf einem „Token“ abzuspeichern. Hier wird dem Datenaustausch zwischen Applikationen eine hohe Bedeutung beigemessen. Deshalb sind zwei häufig in Realisierungen umgesetzte PKCS-Standards die „Cryptographic Message Syntax“ (PKCS # 7) und die „Personal Information Exchange Syntax“ (PKCS # 12). Der PKCS # 7-Standard wird von allen S/MIME-fähigen „E-Mail-Clients“ genutzt, da S/MIME auf Basis von PKCS # 7 ein Format zur Absicherung von elektronischen Nachrichten definiert.

## A.4 Flash-Speicher

Ein wichtiger Faktor im Rahmen einer Produktentwicklung ist die Auswahl der Speichertechnologie. Die Einflussgrößen spielen eine maßgebliche Rolle und beinhalten den Hochtemperaturbetrieb, die Zuverlässigkeit

und Ausführungszeiten der Speicheroperationen. Nach [22] unterscheidet man allgemein zwischen flüchtigen und nichtflüchtigen Speichern. Abbildung A.8 zeigt eine allgemeine Übersicht über verschiedene Speicherarten.

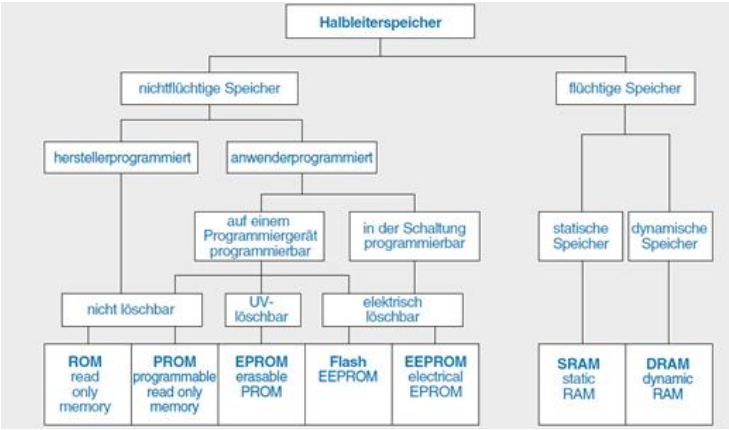


**Abbildung A.8:** Übersicht über verschiedene Speicherarten [22]

Im Embedded- und speziell im automobilen Bereich werden Speicher benötigt, die wenig Strom verbrauchen, temperaturresistent sind und über eine lange Lebensdauer verfügen. Somit haben sich in den letzten 20 Jahren mehrere Generationen von Flash-Speichern etabliert. Dieser Abschnitt beschäftigt sich konkretisiert mit dem Einsatz von Flash-Speichern für die Speicherung von Programmcodes. Hier wird unterschieden, ob es sich um NAND- oder NOR-Architekturen handelt. Diese beiden Architekturen weichen in der Speicherdichte und der Zugriffsgeschwindigkeit voneinander ab. [23]

Diese beiden unterschiedlichen Technologiesätze werden im Folgenden erläutert.

### NOR-orientiert

NOR-orientierte Speicherzellen haben eine parallele Ausrichtung. Der Zugriff erfolgt direkt auf die Speicherzelle, womit die Zugriffszeiten sehr kurz sind. Diese Art der Parallelschaltung ist in Abbildung A.9 dargestellt

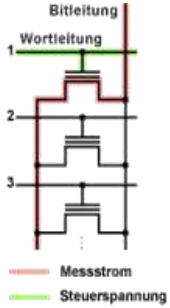


**Abbildung A.9:** Aufbau NOR-Speicher nach [23]

Die Speicherzellen sind in Löschblöcke unterteilt und betragen beispielsweise 128 kBit. Bei einem Löschvorgang werden alle Datenbits auf 1 gesetzt. Der Schreib- und Lesevorgang wiederum ist wortorientiert und kann je nach Datenbusbreite z. B. 32 Bit betragen. Die Lösch- und Schreibzyklen sind demnach begrenzt und können 100k bis 1 Million Zyklen umfassen.

Die Besonderheit liegt darin, dass dann keine Initialisierung der CPU notwendig ist. Sollte der SoC NOR-orientierte Speicher unterstützen, so kann die CPU direkt den NOR-Speicher im eigenen Adressraum abbilden. Damit kann der Code direkt vom NOR-Flash gestartet werden, ohne dass der Code in den Arbeitsspeicher geladen werden muss.

### NAND-orientiert

NAND-orientierte Speicherbausteine sind seriell miteinander verschaltet, wodurch das Lesen und Schreiben nur in Blöcken möglich ist. Durch das Einsparen der Datenleitungen wird auch weniger Platz benötigt, womit sich eine höhere Speicherdichte bildet. Auch die Schreibgeschwindigkeit erhöht sich bei geringerem Stromverbrauch. Abbildung A.10 zeigt den seriell orienterten Speicheraufbau eines NAND-orientierten Speichers. [24]

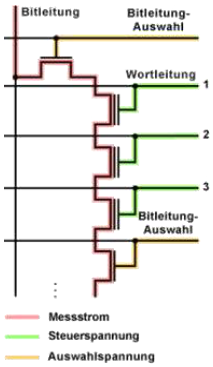


**Abbildung A.10:** Seriell orientierter Speicheraufbau eines NAND-orientierten Speichers nach [24]

Wie beim NOR-orientierten Speicher ist der NAND-orientierte in Löschblöcke unterteilt. Hier liegt die Blockgröße jedoch bei beispielsweise 512 kB. Ein erneutes Löschen eines Blocks setzt alle Bits auf 1. Die Anzahl der Gesamtlöschzyklen ist vergleichsweise gering bei 1k bis 100k Zyklen.

Typischerweise sind pro Speicherzelle zwei verschiedene Zustände möglich. Aufgrund unterschiedlicher Spannungsniveaus durch den Einsatz von Control und Floating Gates kann eine Speicherzelle auch mehrere Zustände annehmen. Man spricht hier von Zellen der Art Single (SLC), Multi (MLC) oder Tripple Level (TLC) (Tripple Level Cell).

Der Lesevorgang ist seitenorientiert und liegt bei 2 oder 4 kBit. Da nicht Byte-weise auf Daten zugegriffen werden kann, ist es notwendig, diese in den Arbeitsspeicher zu laden. Somit muss ein fertiggestellter Code erst in RAM kopiert werden, ehe er ausgeführt werden kann. Hier entsteht ein hohes Risiko für Bitkipper bei Datenübertragungen. Somit ist jede Seite mit einem Fehlerkorrekturcode ausgestattet. SLC-Speicherzellen verwenden einen Hamming Code, der effizient in der eingesetzten Software implementiert ist. Somit lässt sich ein einzelnes Bit innerhalb einer Seite korrigieren. MLC- und TLC-Speicherzellen benötigen komplexere Codes, die bis zu 8-Bit-Fehler pro Seite korrigieren können. Jedoch ist hier eine zusätzliche Hardwareunterstützung vonnöten, um komplexere Fehlererkennungsalgorithmen anwenden zu können. Im Kern werden aber NAND-orientierte Speicher für die Ablage großer Datenmengen verwendet, beispielsweise Kartenmaterial oder Audiodateien für ein Navigationssystem. Der geringe Produktpreis bei hoher Speicherkapazität und wenig Stromverbrauch fördert dessen Einsetzbarkeit.

## A.5 Fehlermöglichkeits- und Einflussanalyse FMEA

Analog zu den ASIL-Klassifikationen setzt sich die RPZ in der FMEA-Betrachtung aus drei verschiedenen Faktoren zusammen, die aus der Wahrscheinlichkeit des Auftretens sowie derjenigen der Entdeckung und der Fehlerschwere bestehen. Nach Hering und Schloske in [63] können durch entsprechende Maßnahmen potentielle Fehler vorausschauend vermieden werden. Das Risikomanagement hilft dabei, Anforderungen der ISO 9001:2015 zu erfüllen. Die analytische Methode einer FMEA gehört daher zum Qualitätsmanagement und zählt zum Standard einer Produktentwicklung. Auch die technische Zuverlässigkeit wird durch die frühzeitige Anwendung einer FMEA im Produktentstehungsprozess erhöht. Da rechtzeitige Vorbeugungsmaßnahmen etabliert werden können, ergeben sich durch die frühzeitige Fehlermöglichkeits- und Einflussanalyse Chancen, Schwachstellen und kritische Bestandteile zu ermitteln. Folgekosten und kostspielige Korrekturen beim Endverbraucher werden somit vermieden bzw. minimiert. Gewonnene Erkenntnisse aus einer FMEA werden bei einer neuen Produktfamilie wiederverwertet. Dies bietet auch weitere Vorteile, wie beispielsweise eine verkürzte Entwicklungszeit.

Im Wesentlichen sind FMEAs in unterschiedliche Bereiche unterteilt und werden je nach Entwicklungszeitpunkt und Anwendung unterschieden. Beispielsweise separiert man zwischen der Konstruktions-FMEA für ein Produkt (Entwicklungs- und Konstruktionsphase) und einer Prozess-FMEA für einen Fertigungsprozess (Produktionsplanungsphase). Die System-FMEA fungiert übergeordnet hinsichtlich des Gesamtsystems und des Zusammenwirkens von Teilsystemen. [63]

**AUTOSAR (AUTomotive Open System ARchitecture)**

Die Architektur und Beschreibung der Software wird in der Regel durch den OEM vorgenommen. Er entscheidet über die Anforderungen und das Zusammenspiel der verschiedenen Softwaresysteme und -komponenten. Durch standardisierte Schnittstellen zwischen verschiedenen Layerorientierten Bereichen lässt sich die Zielapplikation Hardwareunabhängig entwickeln, da die Applikation aus verschiedenen untereinander kommunizierenden Softwarekomponenten kommuniziert. [77, 78]

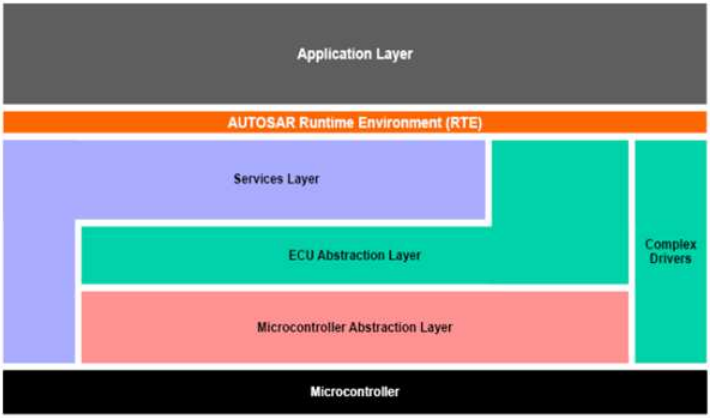


**Abbildung A.11:** AUTOSAR Layered Software Architecture [25]

Die verschiedenen Abstraktions-Layer sind in Abbildung A.11 dargestellt. Als unterste und Hardwareorientierte Schicht steht die einzusetzende Hardwarearchitektur. Die Mittelschicht bis zur Laufzeitumgebung besteht

aus Service-orientierten Layern, Abstraktions-Layern, die über die Ziel-Architektur mit dem SOC verbunden sind. Darauf aufsetzend befindet sich die Laufzeitumgebung, die eigentliche Applikation. Um Sonderfunktionen verschiedener SoCs bereitstellen zu können, werden Treiberstufen entwickelt, die als Complex Drivers bezeichnet werden und in Teilen über die Hardware-, Mittel- und Applikationsschicht reichen. [25]

# A.6 Risikoanalyse

**Tabelle A.1:** Risikoanalyse

| Sicherheitsobjekt | | | | | Angriffspotential | | | | | | | Schadenspotential | Risiko |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Objektname | # | Angriffsszenario | Modell | Sicherheitsrelevante Systemkomponenten | $AP_{Zeit}$ | $AP_{Expertise}$ | $AP_{Information}$ | $AP_{Zugriff}$ | $AP_{Ausrüstung}$ | AP | AP_Name | | |
| Zugriffskontrolle | 1 | Brut-Force-Angriff | Baseline | Baseline.ROOT.APP.SIG | 4 | 7 | 1 | 1 | 0 | 13 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 1 | Brut-Force-Angriff | Modell_1 | Modell_1.ROOT.APP.SIG | 4 | 7 | 1 | 1 | 0 | 13 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 1 | Brut-Force-Angriff | Modell_2 | Modell_2.ROOT.APP.SIG | 4 | 7 | 1 | 1 | 0 | 13 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | Baseline | Baseline.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Zugriffskontrolle | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | Modell_1 | Modell_1.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Zugriffskontrolle | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 2 | Unautorisierte Übernahme einer existierenden Anmeldung | Modell_2 | Modell_2.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Zugriffskontrolle | 3.1 | Manipulation des kryptografischen Schlüssels | Modell_1 | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.1 | Manipulation des kryptografischen Schlüssels | Modell_1 | Modell_1.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.1 | Manipulation des kryptografischen Schlüssels | Modell_2 | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.1 | Manipulation des kryptografischen Schlüssels | Modell_2 | Modell_2.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.1 | Manipulation des kryptografischen Schlüssels | Baseline | ROOT.SCHLÜSSEL | 0 | 7 | 11 | 4 | 7 | 29 | Sehr hoch | Katastrophal | Gering |
| Zugriffskontrolle | 3.2 | Extraktion des kryptografischen Schlüssels | Modell_1 | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.2 | Extraktion des kryptografischen Schlüssels | Modell_1 | Modell_1.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.2 | Extraktion des kryptografischen Schlüssels | Modell_2 | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.2 | Extraktion des kryptografischen Schlüssels | Modell_2 | Modell_2.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 3.2 | Extraktion des kryptografischen Schlüssels | Baseline | ROOT.SCHLÜSSEL | 0 | 7 | 4 | 4 | 7 | 22 | Hoch | Katastrophal | Erhöht |
| Zugriffskontrolle | 4 | Architekturbezogene Sicherheitslücken | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 0 | 11 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 4 | Architekturbezogene Sicherheitslücken | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 0 | 11 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 4 | Architekturbezogene Sicherheitslücken | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 0 | 11 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 4 | Architekturbezogene Sicherheitslücken | Baseline | ROOT.SCHLÜSSEL | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Zugriffskontrolle | 5 | Offene Schnittstellen | Baseline | Baseline.ROOT.APP.SIG | 0 | 3 | 0 | 4 | 4 | 11 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 5 | Offene Schnittstellen | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 3 | 0 | 4 | 4 | 11 | Niedrig | Katastrophal | Hoch |
| Zugriffskontrolle | 5 | Offene Schnittstellen | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 3 | 0 | 4 | 4 | 11 | Niedrig | Katastrophal | Hoch |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Baseline | Baseline.ROOT.INDEX_DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Mittel | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Baseline | Baseline.ROOT.INDEX_ROOT.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Baseline | Baseline.DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.ROOT.INDEX_STRUKTUR.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.STRUKTUR. INDEX_SCHLÜSSE | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.STRUKTUR. INDEX_DATEN.SI | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Mittel | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SI | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_1 | Modell_1.DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.ROOT.INDEX_STRUKTUR | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.STRUKTUR.INDEX_SCHLÜSSE | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.STRUKTUR.INDEX_DATEN | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Mittel | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.STRUKTUR.INDEX_ID.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.SCHLÜSSEL.INDEX_BLOCK | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.SCHLÜSSEL.INDEX_ID.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 1 | Nicht authentifizierte Modifikation der Datensätze | Modell_2 | Modell_2.DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Baseline | Baseline.ROOT.INDEX_DATEN.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Mittel | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Baseline | Baseline.ROOT.INDEX_ROOT.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Baseline | Baseline.ROOT.APP.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Baseline | Baseline.DATEN.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.ROOT.INDEX_STRUKTUR.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.STRUKTUR. INDEX_SCHLÜSSE | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.STRUKTUR. INDEX_DATEN.SI | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Mittel | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SI | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_1 | Modell_1.DATEN.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.ROOT.INSTANZ-SCHLÜSSEL.S | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.ROOT.INDEX_STRUKTUR | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.STRUKTUR.INDEX_SCHLÜSSE | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.STRUKTUR.INDEX_DATEN | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Mittel | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.STRUKTUR.INDEX_ID.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.SCHLÜSSEL.BLOCK-SCHLÜSS | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.SCHLÜSSEL.INDEX_BLOCK | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.SCHLÜSSEL.INDEX_ID.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 2.1 | Tausch des Speicherbausteines | Modell_2 | Modell_2.DATEN.SIG | 0 | 3 | 1 | 10 | 4 | 18 | Mittel | Kritisch | Erhöht |

| Sicherheitsobjekt | | | | | Angriffspotential | | | | | | | Schadenspotential | Risiko |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Objektname | # | Angriffsszenario | Modell | Sicherheitsrelevante Systemkomponenten | $AP_{Zeit}$ | $AP_{Expertise}$ | $AP_{Information}$ | $AP_{Zugriff}$ | $AP_{Ausrüstung}$ | AP | AP_Name | | |
| Datenintegrität | 2.2 | Tausch des Gerätes | Baseline | ROOT.SCHLÜSSEL | 1 | 7 | 4 | 10 | 7 | 29 | Sehr hoch | Katastrophal | Gering |
| Datenintegrität | 3 | Unvollständiges SW-Update | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 3 | Unvollständiges SW-Update | Baseline | Baseline.DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 3 | Unvollständiges SW-Update | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 3 | Unvollständiges SW-Update | Modell_1 | Modell_1.DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Datenintegrität | 3 | Unvollständiges SW-Update | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Katastrophal | Mittel |
| Datenintegrität | 3 | Unvollständiges SW-Update | Modell_2 | Modell_2.DATEN.SIG | 0 | 7 | 0 | 4 | 4 | 15 | Mittel | Kritisch | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Baseline | Baseline.ROOT.APP.SIG | 1 | 7 | 4 | 4 | 4 | 20 | Hoch | Katastrophal | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Baseline | Baseline.DATEN.SIG | 1 | 7 | 4 | 4 | 4 | 20 | Hoch | Kritisch | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Modell_1 | Modell_1.ROOT.APP.SIG | 1 | 7 | 4 | 4 | 4 | 20 | Hoch | Katastrophal | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Modell_1 | Modell_1.DATEN.SIG | 1 | 7 | 4 | 4 | 4 | 20 | Hoch | Kritisch | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Modell_2 | Modell_2.ROOT.APP.SIG | 1 | 7 | 4 | 4 | 4 | 20 | Hoch | Katastrophal | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Modell_2 | Modell_2.DATEN.SIG | 1 | 7 | 4 | 4 | 4 | 20 | Hoch | Kritisch | Erhöht |
| Kryptografische Methoden | 1.1 | Schwache kryptografische Algorithmen | Baseline | ROOT.SCHLÜSSEL | 1 | 7 | 11 | 4 | 4 | 27 | Sehr hoch | Katastrophal | Gering |
| Kryptografische Methoden | 1.2 | Unsicheres Laufzeitverhalten | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Kryptografische Methoden | 1.2 | Unsicheres Laufzeitverhalten | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Kryptografische Methoden | 1.2 | Unsicheres Laufzeitverhalten | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Kryptografische Methoden | 1.2 | Unsicheres Laufzeitverhalten | Baseline | ROOT.SCHLÜSSEL | 0 | 11 | 4 | 4 | 4 | 23 | Hoch | Katastrophal | Erhöht |
| Sichere Laufzeitumgebung | 2 | Manipulierte Applikation | Baseline | Baseline.ROOT.APP.SIG | 0 | 7 | 7,5 | 4 | 4 | 23 | Hoch | Katastrophal | Mittel |
| Sichere Laufzeitumgebung | 2 | Manipulierte Applikation | Modell_1 | Modell_1.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Sichere Laufzeitumgebung | 2 | Manipulierte Applikation | Modell_2 | Modell_2.ROOT.APP.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Katastrophal | Mittel |
| Sichere Laufzeitumgebung | 3 | Manipulierte Datensätze | Baseline | Baseline.DATEN.SIG | 0 | 7 | 7,5 | 4 | 4 | 23 | Hoch | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 3 | Manipulierte Datensätze | Modell_1 | Modell_1.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 3 | Manipulierte Datensätze | Modell_2 | Modell_2.DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Baseline | Baseline.ROOT.INDEX_DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Mittel | Gering |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Baseline | Baseline.ROOT.INDEX_ROOT.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_1 | Modell_1.ROOT.INDEX_STRUKTUR.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Baseline | V2.STRUKTUR.INDEX_SCHLÜSSEL.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Baseline | V2.STRUKTUR.INDEX_DATEN.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Mittel | Gering |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_1 | Modell_1.SCHLÜSSEL.INDEX_BLOCK.SI | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_2 | Modell_2.ROOT.INDEX_STRUKTUR | 0 | 7 | 4 | 1 | 4 | 16 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_2 | Modell_2.STRUKTUR.INDEX_SCHLÜSSE | 0 | 7 | 4 | 1 | 4 | 16 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_2 | Modell_2.STRUKTUR.INDEX_DATEN | 0 | 7 | 4 | 1 | 4 | 16 | Mittel | Mittel | Gering |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_2 | Modell_2.STRUKTUR.INDEX_ID.SIG | 0 | 7 | 4 | 4 | 4 | 19 | Mittel | Kritisch | Erhöht |
| Sichere Laufzeitumgebung | 4 | Injizierte Datensätze | Modell_2 | Modell_2.SCHLÜSSEL.INDEX_BLOCK | 0 | 7 | 4 | 1 | 4 | 16 | Mittel | Kritisch | Erhöht |