HAW HAMBURG

MASTER THESIS
Milena Hippler

Anforderungsanalyse und Bewertung der Technischen Richtlinien des BSI für digitale Gesundheitsanwendungen (DiGAs)

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Milena Hippler

Anforderungsanalyse und Bewertung der Technischen Richtlinien des BSI für digitale Gesundheitsanwendungen (DiGAs)

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Bettina Buth
Zweitgutachter: Prof. Dr. Christian Lins

Eingereicht am: 17. August 2023

Milena Hippler

Thema der Arbeit

Anforderungsanalyse und Bewertung der Technischen Richtlinien des BSI für digitale Gesundheitsanwendungen (DiGAs)

Stichworte

Digitale Gesundheitsanwendungen (DiGA), BSI, Datensicherheit, Mobile Anwendungen, Digitalisierung

Kurzzusammenfassung

Diese Arbeit widmet sich der Analyse und Bewertung der Prüfaspekte gemäß der Technischen Richtlinie TR-03161 des Bundesamtes für Sicherheit in der Informationstechnik (BSI) für Digitale Gesundheitsanwendungen (DiGA). Die Anforderungen an DiGA werden analysiert, Möglichkeiten der Umsetzung nach aktuellem Stand der Technik werden vorgestellt und mittels spezifischer Kriterien bewertet. Die Ergebnisse bieten einen Mehrwert für DiGA-Hersteller, indem die Umsetzungsmöglichkeiten der Anforderungen der Prüfaspekte konkretisiert werden. Auch das BSI kann von den Bewertungen profitieren und anhand dieser Ergebnisse die Prüfaspekte transparenter definieren. Ziel ist es, den Entwicklungsprozess für DiGA-Hersteller zu erleichtern, um die Digitalisierung im Gesundheitswesen positiv zu beeinflussen. Dies geschieht unter der Berücksichtigung der Datensicherheit und der Benutzerfreundlichkeit. Digitale Gesundheitsanwendungen sind ein wesentlicher Teil der Digitalisierung des Gesundheitswesens, denn sie können letztendlich zu einer verbesserten Gesundheitsversorgung und einer an die Patienten angepassten Behandlung führen.

Milena Hippler

Title of Thesis

Analysis of requirements and evaluation of the BSI Technical Guidelines for Digital Health Applications (DiGAs)

Keywords

Digital health applications (DiGA), BSI, data security, mobile applications, digitalization

Abstract

This work is dedicated to the analysis and evaluation of the testing aspects according to the Technical Guideline TR-03161 of the German Federal Office for Information Security (BSI) for Digital Health Applications (DiGA). The requirements for DiGA are analyzed, possibilities of implementation according to the current state of the art are presented and evaluated by means of specific criteria. The results offer added value for DiGA manufacturers by specifying the implementation options for the requirements of the test aspects. The BSI can also benefit from the evaluations and use these results to define the test aspects more transparently. The aim is to facilitate the development process for DiGA manufacturers in order to positively influence digitization in healthcare. This is done with data security and usability in mind. Digital health applications are an essential part of the digitization of healthcare, as they can ultimately lead to improved healthcare and treatment adapted to patients.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADB Android Debug Bridge.

AES Advanced Encryption Standard.

API Application Programming Interface.

ASLR Address Space Layout Randomisation.

BfArM Bundesinsitut für Arzneimittel und Medizinprodukte.

BSI Bundesamt für Sicherheit in der Informationstechnik.

DiGA Digitale Gesundheitsanwendungen.

DiGAV Digitale-Gesundheitsanwendungen-Verordnung.

DLIES Discrete Logarithm Integrated Encryption Scheme.

DLP Diskreten-Logarithmus-Problem.

DSGVO Datenschutz-Grundverordnung.

ECIES Elliptic Curve Integrated Encryption Scheme.

eSE Embedded Secure Element.

FAR Falschakzeptanzrate.

FRR Falschrückweisungsrate.

IPC Interprozesskommunikation.

MAC Message Authentication Code.

PAD Presentation Attack Detection.

PAI Presentation Attack Instrument.

PFS Perfect Forward Secrecy.

REE Rich Execution Environment.

RSA Rivest-Shamir-Adleman.

SSL Secure Socket Layer.

SVDGV Spitzenverband Digitale Gesundheitsversorgung.

TEE Trusted Execution Environment.

TLS Transport Layer Security.

TOTP Time-based One-Time Password.

1 Einleitung

Mobile Anwendungen erleichtern das Leben auf mehrere Weisen und sind ein fest integrierter Bestandteil des Alltags geworden. Die Digitalisierung schreitet in vielen Anwendungsgebieten voran und die Einführung von DiGA spiegelt auch den fortschreitenden Einfluss im Gesundheitswesen wider. Der Spitzenverband Digitale Gesundheitsversorgung (SVDGV) [41] sieht in DiGA das Potenzial, die künftigen Herausforderungen im deutschen Gesundheitswesen (wie beispielsweise den Fachkräftemangel) zu bewältigen. Durch DiGA kann eine bessere Versorgungsqualität erreicht werden und eine Inklusion von Menschen bewirken, die bisher größtenteils aus der Versorgung ausgeschlossen blieben (durch zum Beispiel strukturschwache ländliche Gebiete). DiGA eröffnen neue Behandlungsstrategien und bieten einen Mehrwert dadurch dass sie es ermöglichen, die Bedürfnisse von Patienten sowie ihren Alltag mit der Behandlung von Krankheiten zu verbinden [108].

DiGA müssen je nach Anwendungszweck sensible Gesundheitsdaten erheben, verarbeiten und gegebenenfalls speichern, um ihren Zweck zu erfüllen. Daraus resultieren hohe Anforderungen an den Datenschutz und die Datensicherheit. Die Datenschutz-Grundverordnung (DSGVO) stellt besondere Anforderungen an die Verarbeitung solcher sensiblen Gesundheitsdaten, um die Privatsphäre und die Rechte der betroffenen Personen zu schützen. Die technische Umsetzung muss demnach die Sicherheit der Patientendaten gewährleisten. Denn eine ausnutzbare Sicherheitslücke in einer DiGA kann fatale Folgen haben. Es könnten sensible Gesundheitsdaten offengelegt und kompromittiert werden. Außerdem könnten Daten manipuliert werden, was eine Verletzung der Datenintegrität zur Folge hätte, die zu medizinischen Risiken und Fehlinformationen führen könnte. Aus diesem Grund ist es notwendig, klare Anforderungen an Sicherheitsstandards von DiGA zu definieren, welche dem Stand der Technik entsprechen und dem jeweiligen Anwendungsfall und dessen technischen Bedingungen entsprechend umgesetzt werden können.

Um die Datensicherheit zu gewährleisten, wurde vom Bundesamt für Sicherheit in der Informationstechnik (künftig BSI genannt) die *Technische Richtlinie TR-03161: Anfor-*

derungen an Anwendungen im Gesundheitswesen entwickelt [58]. Diese formuliert Anforderungen an Anwendungen, damit sie als DiGA aus technischer Sicht zugelassen werden. Dabei werden die Schutzziele Vertraulichkeit, Integrität und Verfügbarkeit in der IT-Sicherheit verfolgt.
Herstellerverbände (BVMed, Bitkom, SVDGV) äußern allerdings Bedenken, ob umfangreichen technischen Anforderungen des BSI problemlos von den Herstellern der Anwendungen umsetzbar sind [17] [35]. Es stellt sich also die Frage, ob die Realisierbarkeit dieser Anforderungen im Hinblick auf die Praxis gewährleistet ist.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist es die vom BSI definierten Sicherheitsanforderungen (Prüfaspekte) an eine DiGA zu analysieren, zu bewerten und für die Hersteller Möglichkeiten der Umsetzung vorzustellen, die auf dem aktuellen Stand der Technik basieren. Diese Arbeit soll den DiGA-Herstellern den Prozess erleichtern, die Prüfaspekte des BSI zu erfassen und Recherchearbeiten in Bezug auf Umsetzungsmöglichkeiten abzunehmen. Unter Herstellern sind die Entwickler beziehungsweise Entwicklerteams zu verstehen.
Die Prüfaspekte des BSI weisen teilweise unpräzise Formulierungen auf und lassen Raum für unterschiedliche Interpretationen. Manche Prüfaspekte sind redundant, manche widersprechen sich und einige der Anforderungen sind zu allgemein formuliert, sodass die Erwartungshaltung des BSI nicht deutlich wird. Nur in wenigen Fällen werden konkrete Umsetzungsmöglichkeiten benannt. In den meisten Fällen verweist das BSI auf den Stand der Technik, ohne diesen zu benennen oder es verweist auf andere umfangreiche BSI Richtlinien. Ohne eine ausgiebige Recherche ist es den Herstellern also nicht möglich, die Anforderungen zu erfüllen. Dieser Aufwand soll den Herstellern mittels dieser Arbeit abgenommen werden.
Ziel dieser Arbeit ist es folglich, den Prozess der Erfüllung der Prüfaspekte der TR-03161 für Hersteller zu erleichtern, indem die einzelnen Prüfaspekte übersichtlich gruppiert erklärt, bewertet und mögliche Umsetzungen vorgeschlagen werden. Die Umsetzungsvorschläge dienen nur als Ansatz, was dem Stand der Technik entspricht. Konkrete Implementierungen sind nicht Ziel dieser Arbeit. Sie soll einen Einstiegspunkt bieten, um zu verstehen, was sich hinter den einzelnen Anforderungen verbirgt und was man tun kann, um sie zu erfüllen.
Begleitet wird dieses Ziel durch die Forschungsfrage, inwieweit die Prüfaspekte des BSI-Anforderungskatalogs präzise formuliert sind und wie diese in Hinblick auf die Gesamtheit

der Prüfaspekte umgesetzt werden können. Es wird zudem berücksichtigt, inwieweit die Anforderungen die Benutzerfreundlichkeit der mobilen Anwendung beeinflusst. Denn eine DiGA muss unabhängig von der technischen Affinität oder vom Alter der Patienten verwendet werden können [40].

1.2 Klassifizierung

Es existieren die Technischen Richtlinien TR-03161 [58], welche vom BSI erstellt worden sind. Sie sind das Hauptwerk für die technischen Anforderungen an DiGA. *Teil 1* dieser Richtlinien bezieht sich auf die mobilen Anwendungen und bildet die Grundlage dieser Arbeit.
Dadurch, dass 151 Anforderungen in elf Prüfaspekt-Kategorien behandelt werden, existieren mehrere Themengebiete, die im Bereich Sicherheit in mobilen Anwendungen abgedeckt sein müssen. Neben der TR-03161 wurden für diese Arbeit zunächst weitere technische Richtlinien des BSI priorisiert betrachtet und als Einstiegspunkt für weiterführende Recherchen verwendet. Die Basis bilden also die Prüfaspekte der TR-03161 und anhand dieser Prüfaspekte wurde recherchiert, welche Sicherheitsstandards dem aktuellen Stand der Technik entsprechen, wobei weitere technische Richtlinien des BSI priorisiert behandelt worden sind.
Zum einen ist diese Arbeit so einzuordnen, dass sie als Erweiterung der TR-03161 anzusehen ist und einen Leitfaden für Hersteller präsentiert, um den Prozess der Entwicklung sicherer DiGA zu erleichtern und um erste Interpretationshürden zu überwinden. Zum anderen bewertet sie die Prüfaspekte des BSI auf ihre Präzision und Umsetzbarkeit.

1.3 Aufbau

Nach der Einleitung und der Definition der Forschungsfrage wird im Kapitel 2 eine Einführung in Digitale Gesundheitsanwendungen (DiGA) gegeben. Ziel dieser Einführung ist es ein Grundverständnis davon zu vermitteln, was eine DiGA ist und wie das Zulassungsverfahren abläuft. Außerdem werden in diesem Kapitel die technischen Richtlinien TR-03161 des BSI und die darin enthaltenen Prüfaspekte vorgestellt, welche die Grundlage dieser Arbeit bilden. Nachdem die Grundlagen erklärt worden sind, wird in Kapitel 3 das Konzept erläutert. Darin wird das Vorgehen bezüglich der Gruppierung, Analyse

und Bewertung der Prüfaspekte beschrieben sowie eine Erläuterung des Aufbaus der Bewertungsmatrix und allgemeine Hinweise zum Sprachgebrauch gegeben. Kapitel 4 bildet den Schwerpunkt dieser Arbeit. Für jede Prüfaspekt-Kategorie wird in diesem Kapitel eine Analyse der Anforderungen sowie Umsetzungmöglichkeiten vorgestellt und mittels der Bewertungsmatrix bewertet. In Kapitel 5 werden die Ergebnisse der Arbeit zusammengefasst und es wird ein abschließendes Fazit formuliert. In dem Anhang A sind die Prüfaspekte der *Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen Teil 1: Mobile Anwendungen* [58] hinterlegt.

2 Einführung Digitale Gesundheitsanwendungen (DiGA)

In diesem Kapitel werden die benötigten Grundlagen in Bezug auf Digitale Gesundheitsanwendungen (DiGA) erklärt. Der Abschnitt 2.1 handelt davon, was genau eine DiGA ist und wie diese einzustufen ist. Im Abschnitt 2.2 wird der rechtliche Rahmen von DiGA thematisiert. Nachdem diese Grundlagen geklärt sind, wird in Abschnitt 2.3 das Zulassungsverfahren von DiGA anhand des Leitfaden des Bundesinsitut für Arzneimittel und Medizinprodukte (BfArM) erklärt, um einen Überblick über das allgemeine Verfahren zu schaffen und welche Voraussetzungen nötig sind. Schlussendlich wird die vom BSI erstellte Technische Richtlinie TR-03161 für mobile Anwendungen in dem Abschnitt 2.4 vorgestellt, welche die Grundlage dieser Arbeit bildet.

2.1 Was ist eine DiGA

Eine DiGA ist eine spezielle Art von eHealth-Produkt, welches digitale Technologien nutzt, um die Gesundheitsversorgung zu unterstützen. Es ist zu beachten, dass nicht jede Gesundheitsanwendung eine DiGA ist, denn eine DiGA muss als medizinisches Produkt zertifiziert sein. Für diese Zertifizierung muss beim Bundesinsitut für Arzneimittel und Medizinprodukte ein Prüfverfahren durchlaufen werden, das sogenannte Fast-Track-Verfahren (siehe Abschnitt 2.3). Die gesetzliche Grundlage von DiGA bildet die Digitale-Gesundheitsanwendungen-Verordnung (DiGAV) (siehe Abschnitt 2.2) [68]. Auf Basis dieser gesetzlichen Verordnung hat das BfArM den Leitfaden zum Fast-Track-Verfahren entwickelt [83].
Sobald eine DiGA vom BfArM zugelassen worden ist, kann sie vom Arzt verschrieben und von den Krankenkassen erstattet werden. Der Prozess gestaltet sich folgendermaßen: Der Patient geht zum Arzt und lässt sich die jeweilige DiGA verschreiben. Der Patient meldet sich bei der Krankenkasse, um einen Freischaltcode (DiGA-Code) zu erhalten.

Dieser Freischaltcode wird in die DiGA eingegeben.
DiGA können in Form von mobilen Anwendungen, Web-Anwendungen oder hybriden Anwendungen vorliegen. In Hinblick auf die Informationssicherheit hat das BSI technische Richtlinien definiert (siehe Abschnitt 2.4). Diese technischen Richtlinien sollen sicherstellen, dass DiGA den erforderlichen Sicherheitsstandards für den Schutz von Gesundheitsdaten genügen.
DiGA haben das Potenzial, verschiedene Gesundheitsbereiche abzudecken, von chronischen Krankheiten und psychischer Gesundheit bis hin zu Fitness und Prävention. Folglich besteht das Ziel einer DiGA darin, mithilfe digitaler Technologien die Gesundheitsversorgung zu verbessern. Das wird erreicht, indem sie medizinische Diagnosen, Therapien, Prävention und Überwachung unterstützt und somit zur effektiveren und patientenzentrierten Gesundheitsfürsorge beiträgt [107].

2.2 Rechtlicher Rahmen

Das Bundesministerium für Gesundheit (BMG) hat mit dem Gesetzesentwurf zum Digitalen Versorgungsgesetz (DVG) [68], welches am 19. Dezember 2019 in Kraft getreten ist, und mit der darauf folgenden DiGAV das rechtliche Fundament für DiGA gelegt.
Das DVG zielt darauf ab, die Nutzung digitaler Technologien im Gesundheitswesen zu fördern und die Integration von Digitale Gesundheitsanwendungen sowie anderen digitalen Innovationen in die medizinische Versorgung zu ermöglichen. Die DiGAV konkretisiert die Anforderungen und den Prozess für die Zulassung, Registrierung und Erstattung von DiGA. Sie legt fest, welche Kriterien DiGA erfüllen müssen, um in das DiGA-Verzeichnis aufgenommen und von den gesetzlichen Krankenversicherungen (GKV) erstattet werden zu können. Das BfArM ist die zuständige Behörde, die die Umsetzung der DiGAV überwacht und darauf basierend die Prüfung von DiGA durchführt, um sicherzustellen, dass sie die erforderlichen Qualitäts- und Sicherheitsstandards erfüllen.

2.3 Zulassungsverfahren (Fast-Track-Verfahren)

Das Bundesinsitut für Arzneimittel und Medizinprodukte hat das Fast-Track-Verfahren (Leitfaden siehe [83]) entwickelt. Dieses Verfahren ist das Zulassungsverfahren von DiGA. Nachdem der Hersteller den Antrag beim BfArM gestellt hat, beginnt das Fast-Track-Verfahren, dessen detaillierter Ablauf in Abbildung 2.1 dargestellt ist. Das BfArM prüft

die Anforderungen an die Sicherheit, Funktionstauglichkeit, Qualität, Interoperabilität, Datenschutz und Datensicherheit sowie die positiven Versorgungseffekte (medizinischer Nutzen) [83]. Diese Prüfung soll nach drei Monaten abgeschlossen sein. Folgende Prüfungsergebnisse sind möglich: Die Aufnahme in das DiGA-Verzeichnis, eine vorläufige Aufnahme in das DiGA-Verzeichnis und die Ablehnung beziehungsweise Streichung. Die vorläufige Aufnahme bedeutet, dass der Hersteller nicht in allen Punkten die Prüfung bestanden hat und zu erfüllende Auflagen existieren oder die Evaluierung positiver Versorgungseffekte noch nicht abgeschlossen ist. Die DiGA wird dann für eine sogenannte Erprobungsphase im DiGA-Verzeichnis gelistet. Diese Phase dauert zwölf Monate an. In diesem Zeitraum muss der Hersteller die Auflagen erfüllt haben. Anschließend wird die DiGA entweder dauerhaft in das Verzeichnis aufgenommen oder aus dem Verzeichnis gestrichen.

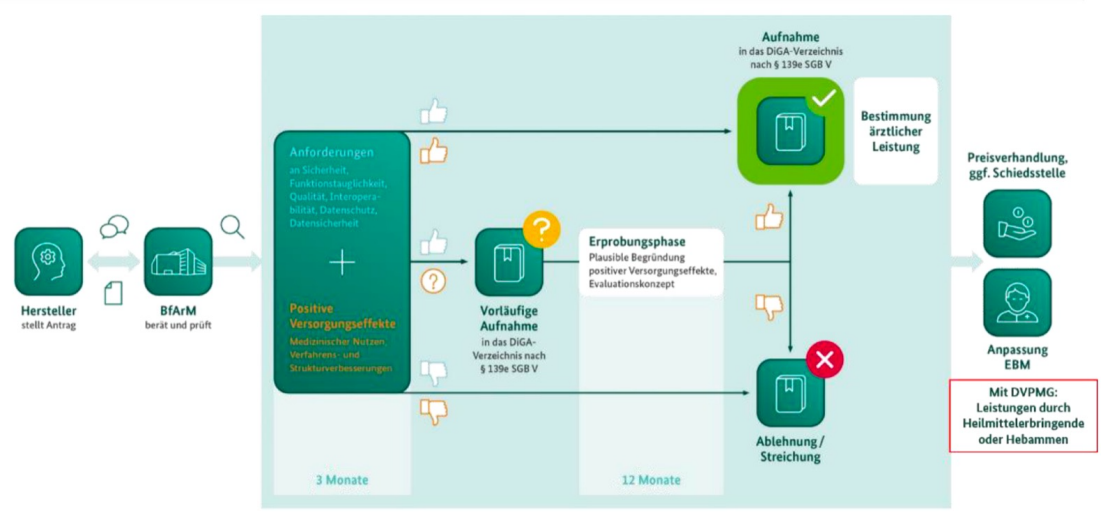

Abbildung 2.1: Ablauf des Fast-Track-Verfahrens [83]

Bevor es zu dem Antrag beim BfArM und somit zu dem Fast-Track-Verfahren kommt, sollten mehrere Anforderungen als Voraussetzung erfüllt werden [82] [65]:

- Zertifizierung als Medizinprodukt abschließen
- Nachweis positiver Versorgungseffekte
- Erfüllung aller Anforderungen aus der Anlage 1 und Anlage 2 der DiGAV [68]

- ISMS-Zertifizierung gemäß ISO 27001 oder BSI IT-Grundschutz (ISO 27001 auf der Basis von IT-Grundschutz)
- Datenschutzzertifizierung nach Art. 42 DSGVO
- Zertifizierung nach TR-03161 (siehe Abschnitt 2.4)

Die Voraussetzung der Zertifizierung nach TR-03161 basiert auf den vom BSI erstellten technischen Richtlinien TR-03161, welche den Schwerpunkt dieser Arbeit bilden. Derzeit prüft noch das BfArM die Einhaltung der dort benannten Anforderungen. Die Prüfung soll ab dem Jahr 2024 vom BSI übernommen werden. DiGA sollen also künftig komplett vom BSI und deren Prüfstellen technisch verantwortet werden. Das BSI hat bereits den TÜVIT [66] als sein erstes Prüflabor benannt, welches die Prüfung durchführen soll.

2.4 Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen – Mobile Anwendungen

Eine DiGA kann in Form einer mobilen Anwendung, einer Web-Anwendung oder einer hybriden Lösung (bestehend aus App- und Web-Anwendung) vorliegen. Zusätzlich spielt auch die Beschaffenheit der zugrundeliegenden Hintergrundsysteme eine Rolle (beispielsweise selbst gehostete Systeme, extern gehostete Systeme und Cloud Computing). Die vom BSI erstellte Technische Richtlinien TR-03161 für Anforderungen an Anwendungen im Gesundheitswesen umfasst drei Teile:

- **Teil 1:** Mobile Anwendungen [58]
- **Teil 2:** Web-Anwendungen [59]
- **Teil 3:** Hintergrundsysteme [60]

Der Fokus dieser Arbeit liegt auf dem *Teil 1: Mobile Anwendungen*. Demnach bildet das vom BSI veröffentlichte Dokument *Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen; Teil 1: Mobile Anwendungen, Version 2.0* die Grundlage für diese Arbeit. Wenn im Folgenden von der TR-03161 die Rede ist, dann bezieht sich dies ausschließlich auf den *Teil 1: Mobile Anwendungen*.

Die TR-03161 soll den DiGA-Herstellern als Richtlinie für die Entwicklung von mobilen Anwendungen dienen. Es ist zu empfehlen, schon bereits in der Planungsphase der Anwendung diese Richtlinien zu berücksichtigen und deren Anforderungen in der Entwicklungsphase umzusetzen. Das BSI definiert in Hinblick auf diese Richtlinien drei Schutzziele: Vertraulichkeit, Integrität und Verfügbarkeit [58].

Der Kern der TR-03161 bilden die Prüfaspekte und die Testcharakteristika. Prüfaspekte sind die spezifischen Anforderungen, die eine mobile Anwendung erfüllen muss, um eine Prüfung durch das BSI oder eine ihrer anerkannten Prüfstellen [66] zu bestehen. Sie umfassen verschiedene technische, funktionale, sicherheitsrelevante und datenschutzbezogene Aspekte, die dazu dienen, die Schutzziele der Anwendungen sicherzustellen und potenzielle Risiken zu identifizieren. Die Prüfaspekte bilden folglich die Grundlage für eine umfassende Beurteilung der mobilen Anwendung in Bezug auf ihre Eignung und Verlässlichkeit im Gesundheitswesen. Testcharakteristika dienen dazu die Umsetzung der Prüfaspekte zu prüfen. Sie erweitern also die Prüfaspekte um ihre Prüftiefe. Testcharakteristika beinhalten außerdem Informationen an den Evaluator, der die Prüfung der mobilen Anwendung durchführen wird, wie die Prüfung zu gestalten ist. Es wird grundlegend zwischen *Check*, also einer Prüfung auf Dokumentenbasis, und *Examine*, einer praktischen Prüfung, unterschieden.
Als Grundlage dieser Arbeit dienen die Prüfaspekte. Die Testcharakteristika wurden nur zu Analyse- und Recherchezwecke hinzugezogen, um die Prüfaspekte vollumfänglich analysieren zu können.

Da in DiGA sensible personenbezogene Daten erhoben und verarbeitet werden, sind die technischen Anforderungen an eine mobile Anwendung im Gesundheitsbereich sehr hoch. Das BSI definiert in der TR-03161 die folgenden elf Prüfaspekt-Kategorien für mobile Anwendungen:

1. Anwendungszweck (siehe Abschnitt 4.1)
2. Architektur (siehe Abschnitt 4.2)
3. Quellcode (siehe Abschnitt 4.3)
4. Drittanbieter-Software (siehe Abschnitt 4.4)
5. Kryptographische Umsetzung (siehe Abschnitt 4.5)
6. Authentifizierung (siehe Abschnitt 4.6)

7. Datensicherheit (siehe Abschnitt 4.7)
8. Kostenpflichtige Ressourcen (siehe Abschnitt 4.8)
9. Netzwerkkommunikation (siehe Abschnitt 4.9)
10. Plattformspezifische Interaktionen (siehe Abschnitt 4.10)
11. Resilienz (siehe Abschnitt 4.11)

Jede dieser Prüfaspekt-Kategorien enthält eine Vielzahl an definierten technischen Anforderungen. Diese müssen laut BSI erfüllt sein, um als DiGA aus technischer Sicht zulässig zu sein. DiGA-Hersteller sollten sich demnach bereits mit Beginn der Planungsphase nach diesen Anforderungen richten.

3 Konzept

In diesem Kapitel wird das Vorgehen und die Einbindung der zuvor geschaffenen Grundlagen vorgestellt, mit denen das folgende Kapitel 4 umgesetzt wird.

Ziel der Arbeit ist es, die vom BSI in der TR-03161 definierten Prüfaspekte detailliert zu erklären, Möglichkeiten der Umsetzung und der Erfüllung vorzustellen und eine abschließende Bewertung anhand definierter Bewertungskriterien in einer Bewertungsmatrix durchzuführen. Dabei wird nicht nur der Aspekt der Umsetzungsmöglichkeiten, sondern auch der Aspekt der Benutzerfreundlichkeit berücksichtigt.
Im Anhang dieser Arbeit befinden sich die Kapitel 3.1.1 - 3.1.11 der TR-03161, welche die einzelnen Definitionen der Prüfaspekte des BSI enthalten. Es empfiehlt sich, diese Definitionen der Prüfaspekte parallel zu dieser Arbeit hinzuzuziehen.

3.1 Vorgehen bei der Analyse und Bewertung der Prüfaspekte

Die in Abschnitt 2.4 vorgestellten elf Prüfaspekt-Kategorien beinhalten jeweils eine Vielzahl weiterer Prüfaspekte. Wenn also in dieser Arbeit von Prüfaspekte gesprochen wird, bezieht sich das auf die jeweiligen Prüfaspekte und nicht nur auf die Oberkategorie. Die Prüfaspekte werden auch oftmals als Anforderungen bezeichnet.
Innerhalb der Prüfaspekt-Kategorien konnte während der Bearbeitung festgestellt werden, dass sich einige Prüfaspekt stark ähneln und sich auf das gleiche Thema konzentrieren. Diese Prüfaspekte wurden gruppiert und werden in einem gemeinsamen Abschnitt vorgestellt. Bei jeder der elf Prüfaspekt-Kategorien wird in dem Kapitel 4 folgendermaßen vorgegangen:

1. Prüfaspekte, die ähnliche Anforderungen definieren, werden gruppiert, um Wiederholungen zu vermeiden und Zusammenhänge besser erkennen und darstellen zu können.

2. Dann wird der jeweilige vom BSI benannte Prüfaspekt kurz erwähnt. Hierbei wird auf den bereits erwähnten Anhang referenziert.

3. Anschließend wird eine detaillierte Erläuterung der Anforderung durchgeführt, Umsetzungsmöglichkeiten vorgeschlagen und Empfehlungen formuliert, welche dem Stand der Technik entsprechen.

4. Für jeden Prüfaspekt oder jede Gruppierung von ähnlichen Prüfaspekten wird unter dem Paragraf *Fazit* eine kurze Zusammenfassung der wesentlichen Ergebnisse durchgeführt. Darin enthalten sind auch die Mindestanforderung der Umsetzung, Hinweise auf mögliche Hindernisse und Risiken sowie eine kurze Bewertung und Einordnung des Prüfaspektes beziehungsweise der Gruppierung.

5. Aus den einzelnen *Fazit*-Bewertungen der jeweiligen Prüfaspekte oder Gruppierung von Prüfaspekte entsteht schlussendlich eine Gesamtbewertung der dazugehörigen Prüfaspekt-Kategorie in Form einer Bewertungsmatrix. Diese dient der Gesamtbewertung, welche in Kapitel 5 ausgewertet wird.

Das Vorgehen bei der Gruppierung wurde gewählt, um Redundanzen zu vermeiden und Bezüge zwischen den Prüfaspekten deutlich zu machen, sodass eine klare Struktur entsteht. Es existieren noch weitere Möglichkeiten der Gruppierung, losgelöst von den Prüfaspekt-Kategorien. Dieser Ansatz wird im Abschnitt 5.2 kurz vorgestellt. In dieser Arbeit fiel die Entscheidung auf die enge Kopplung an die definierten Prüfaspekt-Kategorien, weil der Fokus auf die Umsetzbarkeit und die Bewertung der Präzision der BSI Anforderung gelegt wurde. Aus diesem Grund bietet sich eine enge Kopplung an, um mögliche Probleme besser deutlich zu machen.

3.2 Bewertungsmatrix

Basierend auf dem Fazit der einzelnen Prüfaspekten oder Prüfaspekt-Gruppierungen wird eine Gesamtbewertung in Form einer Bewertungsmatrix am Ende jeder Prüfaspekt-Kategorie durchgeführt. Der Aufbau der jeweiligen Bewertungsmatrix gestaltet sich wie folgt: Die y-Achse der Matrix gibt den jeweiligen Prüfaspekt an. Die x-Achse bildet die Bewertungskriterien ab. Die Bewertungskriterien lauten wie folgt:

- **Aufwand**
  Die Bewertung des Aufwands wird angegeben in *gering*, *mittel* und *hoch*.
  Dieser Bewertungsfaktor berücksichtigt Zeit, Ressourcen und Anstrengungen, die erforderlich sind, um den jeweiligen Prüfaspekt gemäß den Anforderungen umzusetzen. Mit einbezogen wird auch die Komplexität und die Detailliertheit des jeweiligen Prüfaspektes. Eine realistische Einschätzung dieses Aufwands ermöglicht eine effiziente Planung und Ausführung der erforderlichen Schritte.

- **Umsetzbarkeit**
  Die Bewertung der Umsetzbarkeit wird angegeben in *leicht*, *mittel* und *schwer*.
  Dieser Bewertungsfaktor gibt an, inwieweit die Anforderungen des jeweiligen Prüfaspektes tatsächlich erfüllt werden können. Es wird bewertet, ob die Erwartungshaltung des BSI technisch erfüllt werden kann und wie realisierbar die geforderten Maßnahmen sind.

- **Präzision der BSI Anforderung**
  Die Bewertung der Präzision der BSI Anforderung wird angegeben in *eindeutig*, *mittel* und *ungenau*.
  Dieser Bewertungsfaktor bewertet die Klarheit, Eindeutigkeit und Detailgenauigkeit der BSI Anforderungen. Eine genaue Formulierung erleichtert die korrekte Interpretation und Umsetzung der Vorgaben, wodurch potenzielle Missverständnisse vermieden werden können.

- **Höhe der Priorität im Gesamtprojekt**
  Die Bewertung der Höhe der Priorität im Gesamtprojekt wird angegeben in *niedrig*, *mittel* und *hoch*.
  Dieser Bewertungsfaktor bewertet die Relevanz eines Prüfaspektes im Kontext des Gesamtprojekts. Diese Bewertung ermöglicht eine angemessene Ressourcenverteilung und die Fokussierung auf jene Prüfaspekte, die einen größeren Einfluss auf die Sicherheit der Anwendung haben.

Die Bewertungen basieren auf der Analyse der jeweiligen Prüfaspekte sowie deren Fazit. Die Bewertung ist so gestaltet, dass nur die jeweilige Prüfaspekt-Kategorie betrachtet und bewertet wird ohne Prüfaspekte anderer Kategorien mit in die Bewertung einzubeziehen. Die Bewertung soll in erster Linie den Herstellern dienen, aber auch das BSI kann aus dieser Arbeit und den Bewertungen einen Mehrwert ziehen.

Jede der angegebenen Bewertungen erfolgt intern auf einer Skala von ein bis neun (Quantifizierung). Wobei der untere Bewertungsbereich (*gering*, *leicht*, *eindeutig* und *niedrig*) von den Werten eins bis drei abgedeckt wird, der mittlere Bewertungsbereich (*mittel*) von den Werten vier bis sechs und der obere Bewertungsbereich (*hoch*, *schwer* und *ungenau*) von den Werten sieben bis neun. Die Wahl der Darstellung der Bewertungbereiche fiel auf diese qualitative Zuordnung, um eine leicht verständliche Bewertung zu gewährleisten. Die Summe der vergebenen Bewertungswerte wird dividiert durch die Anzahl der bewerteten Prüfaspekte. Der daraus resultierende Mittelwert bildet das Ergebnis der Gesamtbewertung der Prüfaspekt-Kategorie. Diese Vorgehensweise ermöglicht es, den Bewertungsspielraum für jeden einzelnen Prüfaspekt zu erweitern und eine flexiblere Einschätzung vorzunehmen. Auf diese Art findet in jeder Bewertung eine feinere Abstufung statt, die die resultierende Gesamtbewertung nuancierter beeinflusst.

3.3 Allgemeine Anmerkungen

Diese Arbeit bezieht sich auf mobile Anwendungen. Es existieren für sowohl Android als auch iOS DiGA. In manchen Fällen wird zwischen den Plattformen unterschieden, wenn keine explizite Unterscheidung (durch den Hinweis Android oder iOS) vorhanden ist, gilt es für beide. Beispielsweise wenn von dem *App Store* die Rede ist ohne expliziten Hinweis welches spezifische Betriebssystem gemeint ist, dann ist davon auszugehen, dass sich die Aussage sowohl auf den Google Play Store als auch auf den Apple App Store bezieht. Diese Entscheidung wurde getroffen, um Redundanzen zu vermeiden und den Lesefluss zu erleichtern.

Hinweis zum Gendern In dieser Arbeit wird zur Verbesserung der Lesbarkeit und des Schreibflusses vorwiegend die maskuline Form verwendet, um auf Personen zu verweisen. Es sei betont, dass diese Verwendung keinerlei Geschlechterdiskriminierung beabsichtigt. Sämtliche verwendeten maskulinen Bezeichnungen gelten als geschlechtsneutral und schließen alle Geschlechter gleichermaßen ein.

4 Detaillierte Analyse und Bewertung der Prüfaspekte

In dem vorherigen Kapitel 3 wurden bereits die Grundlagen in Bezug auf das Vorgehen, die Bewertungsmatrix sowie allgemeine Hinweise beschrieben, die für dieses Kapitel benötigt werden.

Die für dieses Kapitel relevanten Teile der *Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen, Teil 1: Mobile Anwendungen in der Version 2.0* [58] befinden sich im Anhang A dieser Arbeit.
Es ist wichtig, den referenzierten Anhang für jeden Prüfaspekt, der behandelt wird, hinzuzuziehen. Dort ist die genaue Formulierung des BSI für den jeweiligen Prüfaspekt hinterlegt. Diese Formulierung ist die Basis für das weitere Vorgehen. Das allgemeine Vorgehen bei der Analyse und Bewertung der Prüfaspekt-Kategorien wurde bereits im Konzept in Kapitel 3 vorgestellt.

4.1 Anwendungszweck

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Anwendungszweck fallen.
Die darin enthaltenen Prüfaspekte decken Anforderungen an die Verarbeitung personenbezogener Daten in Bezug auf den rechtmäßigen Zweck der Anwendung ab.

4.1.1 Prüfaspekte O.Purp_1 und O.Purp_2

Diese beiden Prüfaspekte lassen sich zusammenfassen und gemeinsam bearbeiten. Beide Anforderungen zielen es darauf ab, dass nur Daten erhoben werden, welche dem rechtmäßigen Zweck der Anwendung dienen. O.Purp_1 (vgl. Anhang A.1) bezieht sich primär

auf die Nutzersicht, also dass der Nutzer über die Verarbeitung seiner Daten ausreichend und umfänglich informiert wird und O.Purp_2 (vgl. Anhang A.1) knüpft daran an und stellt sicher, dass nur notwendige Daten gesammelt und verarbeitet werden.
Im Detail bedeutet das, dass es notwendig ist, den Nutzer darauf hinzuweisen, welchen genauen Zweck die Anwendung erfüllt und welche personenbezogenen Daten gesammelt und verarbeitet werden. Es ist wichtig sicherzustellen, dass dieser Hinweis in einer benutzerfreundlichen Sprache formuliert wird, damit jeder Nutzer den Zweck und die Datenverarbeitung verstehen kann.
Diese Informationen müssen dem Nutzer vor der Installation der mobilen Anwendung vorliegen. Folgende Ansätze der Bereitstellung der Informationen können für die Umsetzung von O.Purp_1 gewählt werden, wobei der erste Punkt notwendig ist:

1. Wie das BSI für diese Anforderung bereits empfohlen hat, sollte der Hersteller die rechtmäßigen Zwecke der Anwendung und die Verwendung von personenbezogenen Daten in der Beschreibung im App Store klar und deutlich angeben. Dies sollte in einfachen und verständlichen Worten erfolgen, um den Nutzern eine klare Vorstellung davon zu vermitteln, welche Daten erfasst und wie sie verwendet werden [48][26]. So kann der Nutzer sich direkt vor dem Download und Installation der mobilen Anwendung ein genaues Bild machen.

2. Mittels Datenschutzerklärung. Die Datenschutzerklärung sollte auch vor der Installation aufrufbar sein, durch die Website des Herstellers oder über einen Link in der App Store Beschreibung. Die Datenschutzerklärung sollte transparente Informationen über die Art der gesammelten personenbezogenen Daten, den Zweck der Datenerhebung, die Verarbeitungsmethoden, die Speicherfristen und andere relevante Informationen enthalten.

3. Da der potenzielle Nutzer die DiGA vom Arzt verschrieben bekommt, ist ein üblicher Weg, dass der Patient vom Arzt zunächst einen Flyer erhält, mit dem er sich erstmalig mit der DiGA auseinandersetzen kann. Bereits in diesem Flyer kann ein Hinweis der Datenverarbeitung angerissen werden.

Die Dokumentation der Daten, die zur Erfüllung von O.Purp_2 erforderlich sind, sollte möglichst folgende Informationen über die Daten enthalten [42]:

- Datentyp

- Zweck innerhalb der Anwendung
- Speicherort (Land, EU, Cloud, lokal auf einem Endgerät etc.); ggf. auch Hinweis darauf, ob dieser Datentyp an Dienstleister übertragen wird
- Speicherdauer / Löschfrist
- Methode der Verschlüsselung (sofern verschlüsselt)
- Rechtsgrundlage nach DSGVO

Fazit Zwingend notwendig ist für O.Purp_1 die Beschreibung der Zwecke und personenbezogenen Daten im App Store, da dies genau auch vom BSI benannt wird. Die Datenschutzerklärung ist nicht verpflichtend, um diesen Punkt zu erfüllen, aber ist durchaus erwünscht. Ein Hinweis im Flyer ist ebenfalls optional, vermittelt dem Nutzer aber Transparenz und schafft beim Nutzer ein Bewusstsein dafür. Zusammenfassend ist nur der App Store obligatorisch für das Bestehen der Prüfung.

Die Bereitstellung der Informationen ist kein großer Aufwand, insofern während der Entwicklung der Anwendung ausreichend dokumentiert worden ist. Das größte Hindernis liegt in den Phasen der Projektplanung und der Umsetzung. Bereits bei der Planung der Anwendung darf der Fokus auf den Zweck der Anwendung nicht verloren gehen. DiGA müssen so funktional und auf den Anwendungszweck bezogen sein. Also empfiehlt es sich bereits zum Beginn des Projektes genau zu überlegen, welche personenbezogene Daten gebraucht werden und an welcher Stelle man die Erhebung der Daten minimieren kann. In der Phase der Umsetzung sollte immer wieder geprüft und dokumentiert werden, welche Daten erhoben werden und ob diese wirklich notwendig sind.

4.1.2 Prüfaspekte O.Purp_3, O.Purp_4, O.Purp_5 und O.Purp_6

Diese Anforderungen beziehen sich auf Nutzereinwilligungen. Sie behandeln verschiedene Aspekte, die mit der Einholung der Verwaltung und dem Widerruf der Einwilligung in Bezug auf die Erfassung und Verarbeitung personenbezogener Daten in einer Anwendung zusammenhängen.
Zunächst muss die Anwendung eine aktive und eindeutige Einwilligungserklärung des Nutzers einholen, bevor personenbezogene Daten erfasst oder verarbeitet werden (O.Purp _3, s. A.1). Die Einwilligung des Nutzers ist unerlässlich, um sicherzustellen, dass die

Datenerfassung und -Verarbeitung im Einklang mit den Datenschutzbestimmungen stehen. Der Prüfaspekt O.Purp_4 (vgl. Anhang A.1) betont, das Daten, denen der Nutzer nicht zugestimmt hat, auf keinen Fall erhoben werden dürfen.
Die Einwilligungserklärung muss dem Nutzer vor dem Zugriff auf die Funktionalitäten der Anwendung, die eine Erfassung oder Verarbeitung personenbezogener Daten erfordern, präsentiert werden. Es bietet sich also an, direkt nach dem Starten der Anwendung die Einwilligungserklärung in Form eines Dialogfelds, eines Pop-ups oder einer separaten Bildschirmseite anzuzeigen. Es ist wichtig, dass die Einwilligungserklärung deutlich und verständlich formuliert ist, sodass der Nutzer die Folgen seines Einverständnisses verstehen kann. Zusätzlich sollte die Einwilligungserklärung so gestaltet sein, dass der Nutzer aktiv zustimmen muss, anstatt dass die Einwilligung automatisch angenommen wird. Der Nutzer sollte die Möglichkeit haben, die Einwilligung abzulehnen oder den Zugriff auf bestimmte Daten zu beschränken, falls dies von der Anwendung unterstützt wird [25].

Der Prüfaspekt O.Purp_5 (vgl. Anhang A.1) fordert, dass der Nutzer jederzeit die Möglichkeit haben muss die erteilte Einwilligung zu widerrufen. Im Fall eines Widerrufs sollte der Nutzer darauf hingewiesen werden, dass der Widerruf Folgen für die Nutzung der Anwendung haben kann. Weiterführend ist der Widerruf einer Einwilligung mit der Verpflichtung des Herstellers verbunden, die betreffenden personenbezogenen Daten zu löschen oder zu sperren [58]. Hierfür existieren folgende exemplarische Ansätze [36] [84]:

- **Datenlöschung**
  Der Hersteller kann die personenbezogenen Daten des Nutzers physisch aus den Datenbanken oder Speichersystemen entfernen. Es ist wichtig sicherzustellen, dass alle Kopien der Daten gelöscht werden, einschließlich Backups oder Datenspiegelungen.

- **Datenanonymisierung**
  Anstelle der vollständigen Löschung kann der Hersteller die Daten des Nutzers anonymisieren. Es werden alle Identifikationsmerkmale entfernt, sodass die Daten nicht mehr einer bestimmten Person zugeordnet werden können

- **Datenblockierung oder Sperrung**
  Eine weitere Möglichkeit besteht darin, die Daten des widerrufenden Nutzers zu sperren oder zu blockieren. Dadurch wird verhindert, dass die Daten weiterhin für Verarbeitungszwecke verwendet werden, bleiben jedoch im System erhalten. Dies kann nützlich sein, um sicherzustellen, dass rechtliche Anforderungen erfüllt werden, zum Beispiel wenn es Aufbewahrungspflichten gibt oder die Daten für die Abwicklung von Rechtsansprüchen benötigt werden.

Es ist ratsam, Einwilligungen als Abschnitt im Menü der Anwendung oder als Unterpunkt in den Einstellungen zu integrieren, sodass es benutzerfreundlich und intuitiv zu finden ist. Zum einen könnte der Nutzer in diesem Bereich die Möglichkeit haben, den Widerruf seiner erteilten Einwilligung durchzuführen. Zum anderen könnte der Nutzer in diesem Bereich eine Übersicht oder einen Auszug der Einwilligungserklärung finden, in dem aufgeführt ist, welchen Datenkategorien oder spezifischen Verarbeitungszwecken er zugestimmt hat. Dies ermöglicht es dem Nutzer, seine erteilten Einwilligungen zu überprüfen und sich bewusst zu sein, welche Daten in der Anwendung erfasst und genutzt werden. Der Nutzer sollte an dieser Stelle auch die Möglichkeit haben, eine Historie seiner Einwilligungen und gegebenenfalls Widerrufe anfordern zu können.
Aus diesem Grund ist es zwingend notwendig, ein Einwilligungsverzeichnis seitens des Herstellers zu führen (O.Purp_6, s. A.1). Ein solches Verzeichnis sollte folgende Informationen enthalten [25] [34]:

- **Nutzeridentifikation**
  Eine eindeutige Kennung oder Benutzer-ID, die den jeweiligen Nutzer eindeutig identifiziert.
- **Zeitpunkt der Einwilligung**
  Das Datum und die genaue Uhrzeit, zu der der Nutzer seine Einwilligung erteilt hat.
- **Einwilligungszweck**
  Eine Beschreibung der konkreten Datenverarbeitungszwecke, für die der Nutzer seine Einwilligung gegeben hat.
- **Datenkategorien**
  Eine Aufschlüsselung der Arten von personenbezogenen Daten, denen der Nutzer zugestimmt hat, erfasst oder verarbeitet zu werden.
- **Dauer der Einwilligung**
  Informationen darüber, wie lange die Einwilligung des Nutzers gültig ist oder ob sie zeitlich begrenzt ist. Es kann beispielsweise angegeben werden, ob die Einwilligung bis zum Widerruf des Nutzers gültig bleibt oder ob sie für einen festgelegten Zeitraum gilt.
- **Status der Einwilligung**
  Der aktuelle Status der Einwilligung, ob sie aktiv ist oder vom Nutzer widerrufen wurde.

Fazit Der Inhalt einer Einwilligungserklärung sowie wann die Einwilligung am besten abgefragt werden sollte, wurde bereits ausführlich erklärt. Daran sollten sich Hersteller auf jeden Fall halten. Das gleiche gilt für das Einwilligungsverzeichnis und wie der Zugang zu den Informationen benutzerfreundlich in die Anwendung eingebaut werden kann. In Bezug auf die Anforderungen an den Widerruf einer Einwilligung und der daraus resultiernden Verpflichtung, die betreffenden personenbezogenen Daten zu löschen oder zu sperren, wurden Umsetzungsmöglichkeiten vorgestellt. Welcher Ansatz für welche Anwendung und für welchen Datensatz geeignet ist, entscheidet der Zweck und die Beschaffenheit der Daten. Eine Datenlöschung wird in jedem Fall bei einer Prüfung nicht bemängelt werden. Wenn es also möglich ist, ist dieser Weg zu priorisieren.

4.1.3 Prüfaspekte O.Purp_7, O.Purp_8 und O.Purp_9

In beiden Prüfaspekten O.Purp_7 und O.Purp_8 (vgl. Anhang A.1) liegt der Fokus auf dem Umgang mit Dritten, sei es durch die Verwendung von Frameworks und Bibliotheken oder durch die Weitergabe sensibler Daten.
In Abschnitt 4.4 wird detaillierter auf den Umgang mit Frameworks und Bibliotheken von Dritten eingegangen; hier werden demnach die Punkte nur in Bezug auf die Anforderungen an den Anwendungszweck erörtert. In Prüfaspekt O.Purp_9 (vgl. Anhang A.1) wird Bezug auf sensible Daten genommen, welche auch von O.Purp_8 behandelt werden. Für diese Anforderungen wird wieder die in 4.1.1 beschriebene Dokumentation der rechtmäßigen Zwecke betrachtet, denn diese gelten auch für Funktionen, die von Frameworks und Bibliotheken von Dritten bereitgestellt werden. Dritte stellen in der Datenverarbeitung ein Sicherheitsrisiko dar, weshalb genau abgewägt werden muss, ob der Einsatz von Funktionen Dritter zwingend notwendig ist oder ob nicht eine eigene Lösung möglich wäre. Wenn Frameworks oder Bibliotheken von Dritten verwendet werden, müssen nicht für den Zweck benötigte Funktionen deaktiviert werden. Es sollte also bei Nutzung Frameworks und Bibliotheken Dritter darauf geachtet werden, dass diese konfigurierbar sind oder modular aufgebaut sind, sodass nicht benötigte Funktionen deaktiviert oder deren Module gar nicht erst geladen werden.
In Bezug auf die Weitergabe von sensiblen Daten ist darauf zu verzichten, es sei denn, es ist für den primären Zweck der Anwendung erforderlich und nicht mit eigenen Mitteln lösbar ist. Außerdem muss darauf geachtet werden, dass sensible Daten nicht in dem öffentliche Teil des Dateisystems abgelegt werden. Der öffentliche Teil des Dateisystems bezieht sich auf den Bereich eines Dateisystems, der für den allgemeinen Zugriff freigege-

ben ist. Sensible Daten dürfen weder in der Anwendung selbst noch in Bezug auf Dritte auf dem Display dargestellt werden, außer wenn es dem primären Zweck dient. Weiterführend soll keinerlei visueller Zugang zu sensiblen Daten möglich sein um Bedrohungen wie Schulter-Surfen zu verhindern. Hierbei gelangt der Angreifer an sensible Daten, indem sie auf der Anwendung dargestellt werden und er sie mitlesen kann [58].
Das BSI sieht außerdem vor, dass 'die Anwendung den Nutzer uber die Konsequenzen einer eventuellen Weitergabe von Anwendungsdaten vollumfänglich informieren muss und sein Einverständnis einholen' [58]. Um sicherzustellen, dass ein Nutzer seine Zustimmung bewusst und freiwillig gibt, sollte die Anwendung ein Einverständnis (OPT-IN) einholen. Das bedeutet, dass der Nutzer aktiv zustimmen muss, dass seine Daten weitergegeben werden dürfen. Dies könnte beispielsweise durch das Anzeigen der Einwilligungserklärung 4.1.2 oder das Abhaken eines Kontrollkästchens erfolgen.

Fazit Insofern es sich umsetzen lässt und nicht zu viel extra Aufwand bedeutet, ist es ratsam, so wenig auf Dritte angewiesen zu sein wie möglich. Dies gilt vor allem in Bezug auf die Weitergabe von sensiblen Daten. Außerdem ist es nicht immer möglich, nicht zweckgebundene Funktionen von Dritten zu deaktivieren. Das allein stelle ein Risiko dar, die Prüfung in diesem Punkt nicht zu bestehen, denn es sind nur zweckgebundene Funktionen erlaubt, alle anderen müssen deaktiviert werden. Eine ausführliche Abwägung und Begründung, warum nicht auf Dritte verzichtet werden konnte, muss für die Prüfung erstellt werden. Wenn Dritte unerlässlich sind, sollte der Hersteller in 4.1.1 für die Erhebung und Verarbeitung personenbezogener Daten die rechtmäßigen Zwecke klären und zu dokumentieren.
Hinweis: Für diesen Prüfaspekt ist unbedingt der Prüfaspekt O.TrdP_6 (siehe Abschnitt 4.4.1) zu beachten. Dort erlaubt das BSI die Weitergabe sensibler Daten an Drittanbieter nur, wenn es dem Zweck der Verschlüsselung dient. Andere primäre Zwecke werden nicht zugelassen. Also muss die dortige Empfehlung für diesen Prüfaspekt berücksichtigt werden.

4.1.4 Bewertung Anwendungszweck

Bewertungsaspekt → Prüfaspekt ↓	Aufwand	Umsetzbarkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
O.Purp_1, O.Purp_2	hoch	mittel	eindeutig	hoch
O.Purp_3, O.Purp_4, O.Purp_5, O.Purp_6	gering	leicht	eindeutig	niedrig
O.Purp_7, O.Purp_8, O.Purp_9	hoch	schwer	mittel	hoch
Gesamtbewertung	🟡	🟡	🟢	🟡

Tabelle 4.1: Bewertung Anwendungszweck

4.2 Architektur

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Architektur fallen. Die darin enthaltenen Prüfaspekte decken Anforderungen an die strukturelle Gestaltung und Organisation von DiGA ab. Dieses Kapitel legt Kriterien fest, um eine robuste und skalierbare Architektur sicherzustellen, die die Integrität, Vertraulichkeit und Verfügbarkeit der Gesundheitsdaten gewährleistet und eine langfristige Wartbarkeit der Anwendung fördert.

4.2.1 Prüfaspekte O.Arch_1, O.Arch_5, O.Arch_8 und O.Arch_9

Der Prüfaspekt O.Arch_1 (vgl. Anhang A.2) legt fest, dass Security ein fester Bestandteil des Softwareentwicklungs- und Lebenszyklus der Anwendung sein soll. Leider ist diese Beschreibung sehr abstrakt formuliert, denn Security ist ein weitläufiger Begriff und bis auf den Hinweis, dass ein Vergleich mit den Betriebssystem spezifischen Security Standards durchgeführt werden sollte, gibt das BSI keinerlei Informationen, was explizit als Minimum erwartet wird.
Um diesem Prüfaspekt gerecht zu werden, verweist das BSI also auf zwei Quellen: Das 'iOS Security Framework' für iOS-Anwendungen [50] und das 'Security for Android Developers' für Android-Anwendungen [27]. Diese Quellen bieten Richtlinien und Best Prac-

tices für die Sicherheitsarchitektur von mobilen Anwendungen auf den jeweiligen Plattformen. Auf Basis dieser Information wurde eine Recherche durchgeführt, was genau unter Security als fester Bestandteil der Softwareentwicklung und dem Lebenszyklus der Anwendung zu verstehen ist. Folgende Best-Practices für Security in mobilen Anwendungen sollten berücksichtigt werden [27] [50] [3] [10] [97]:

- **Sichere Datenübertragung**
  Umsetzung von Netzwerk- und Kommunikationssicherheit (HTTPS, TLS/SSL). Anforderungen werden in dem Prüfaspekt in Abschnitt 4.5.2 genauer betrachtet.
- **Authentifizierung und Autorisierung**
  Implementierung einer sicheren Benutzerauthentifizierung und Autorisierung, um sicherzustellen, dass nur autorisierte Benutzer auf geschützte Funktionen und Daten zugreifen können. Anforderungen werden in dem Prüfaspekt 4.6 genauer betrachtet.
- **Speicher-Management**
  Verschlüsselung von gespeicherten Daten, Festlegung von Berechtigungen für den Zugriff auf gespeicherte Daten, um sicherzustellen, dass nur autorisierte Benutzer oder Prozesse auf die Daten zugreifen können. Präzisere Anforderungen sind dem Prüfaspekt Datensicherheit in Abschnitt 4.7 zu entnehmen sowie dem Abschnitt 4.2.2.
- **Code Obfuscation**
  Der Prozess der Code Verschleierung ist wichtig, um Reverse Engineering zu vermeiden. Der Quellcode wird somit vor Decompilern und menschlicher Analyse geschützt. Mehr zu Verschleierungsmaßnahmen ist dem Abschnitt 4.11.3 zu entnehmen.
- **Datenverschlüsselung**
  Anforderungen an die Datenverschlüsselung können dem Prüfaspekt kryptographische Umsetzung in Abschnitt 4.5 entnommen werden.
- **Nutzung vertrauenswürdiger Bibliotheken**
  Wurden bereits in Abschnitt 4.1.3 beschrieben und werden in Abschnitt 4.4 weiter konkretisiert.

Weitere Punkte, die bei der Planung und Entwicklung berücksichtigt werden sollten, sind Berechtigungsmanagement, Data Minimization, reguläres Updaten und Testen, Session Management, Logging und User Input Validation [3] [97]. Viele dieser Punkte werden als

Anforderungen in folgenden Prüfaspekten erwähnt und werden aus diesem Grund hier nun nicht weiter behandelt.

In Bezug auf O.Arch_5 (vgl. Anhang A.2) ist darauf zu achten, dass vorgestellte Sicherheitsfunktionen (Authentifizierung, Autorisierung, Input Validation) auf allen Außenschnittstellen und API-Endpunkten der Anwendung implementiert werden müssen.
Die Anforderung der Sicherheitsfunktionen der Schnittstellen in Bezug auf Input Validation wird durch die Anforderung O.Arch_8 (vgl. Anhang A.2) genauer erläutert. In dieser Anforderung geht es um interpretierten Code. Interpretierter Code bezieht sich auf Code, der zur Laufzeit einer Anwendung interpretiert und ausgeführt wird, im Gegensatz zu vorkompiliertem Code, der vor der Ausführung übersetzt wird. Hierunter fallen Benutzereingaben (Tastatureingaben, Mausklicks oder Touch-Gesten). Durch Input Validation sollen Benutzereingaben geprüft werden. Nur wenn es dem primären Zweck der Anwendung dient (muss ausreichend begründet sein), darf ein Zugriff mittels interpretierten Code auf Speicher oder Nutzerdaten stattfinden.
Gemäß der Anforderung O.Arch_9 (vgl. Anhang A.2), muss der Nutzer die Möglichkeit haben dem Hersteller jegliche Sicherheitsprobleme über einen verschlüsselten Kanal (siehe Abschnitt 4.9.1) zu melden.

Generell gilt der Ansatz, dass Sicherheitsaspekte von Anfang an in den Planungs- und Entwicklungsprozess integriert werden sollten und kontinuierlich während des gesamten Lebenszyklus der Anwendung berücksichtigt werden müssen. Das bedeutet, dass der Hersteller auf dem neusten Stand der Technik bleiben sollte und anhand dessen die Anwendung auf zukünftig weiterentwickeln muss (Security Updates).

Fazit Da diese Anforderung sehr grob und unspezifisch formuliert wurde, kann nicht genau erschlossen werden, was von dem Hersteller erfüllt werden muss, um diese Anforderung zu erfüllen. Die vorgestellten Möglichkeiten, Security umzusetzen, sind die Ergebnisse der Recherche einiger Quellen. Die hier genannten Security-Anforderungen werden in späteren Prüfaspekten noch genauer behandelt. Aus diesem Grund lohnt es sich, zuerst diese bei der Planung zu berücksichtigen und dann unter diesem Prüfaspekt zu validieren, ob alles berücksichtigt worden ist. Es handelt sich bei diesen Prüfaspekten folglich mehr um eine Einleitung für die folgenden.

4.2.2 Prüfaspekt O.Arch_2 und O.Arch_4

Beide Prüfaspekte konzentrieren sich auf den Umgang mit sensiblen Daten. Der Prüfaspekt O.Arch_2 (vgl. Anhang A.2) geht auf den Datenlebenszyklus ein und O.Arch_4 (vgl. Anhang A.2) fordert, dass Backups und Cloud-Backups keine sensiblen Daten im Klartext beinhalten dürfen. Für O.Arch_4 bedeutet das, dass sensible Daten nur ausreichend verschlüsselt in Backups gespeichert werden dürfen, Informationen zu geeigneten Verschlüsselungen können den Anforderungen der kryptographischen Umsetzung im Abschnitt 4.5 entnommen werden.
In Bezug auf O.Arch_2 muss die Architektur der Anwendung die sichere Erhebung, Verarbeitung, Speicherung und Löschung dieser sensiblen Daten in einem Datenlebenszyklus abbilden. Es existieren viele verschiedene Ansätze, ein Datenlebenszyklus-Modell umzusetzen; Vorschläge können den Quellen [98], [114] und [11] entnommen werden. Welches Modell geeignet ist, ist von Anwendung zu Anwendung unterschiedlich. Allgemein ist es das Ziel, eine Abfolge von Phasen beziehungsweise Aktivitäten darzustellen [115]. Aus diesem Grund sollten folgende Punkte bei der Erstellung des Zyklus berücksichtigt werden, die den Lebenszyklus von sensiblen Daten beschreiben [112] [74]:

1. **Datenerhebung**
   Diese Phase umfasst das Sammeln von sensiblen Daten. Hierbei muss beachtet werden, dass die Erhebung gemäß den geltenden Datenschutzbestimmungen und Einwilligungen des Nutzers erfolgt (siehe Abschnitt 4.1.2). Es muss sichergestellt werden, dass nur erforderliche Daten erfasst und sicher und geschützt gespeichert werden.

2. **Datenspeicherung**
   Wo werden die Daten gespeichert, wie werden sie verschlüsselt. Ergänzend können hier Angaben zu getroffenen Sicherheitsmaßnahmen wie Zugriffskontrollen und Auditing, um die Integrität und Verfügbarkeit der Daten zu gewährleisten, beschrieben werden.

3. **Datenverarbeitung und Datenaustausch**
   Die Architektur der Anwendung sollte sicherstellen, dass Zugriffskontrollen implementiert sind, sodass nur autorisierte Personen oder Systeme auf die sensiblen Daten zugreifen und sie verarbeiten können.

4. **Datenlöschung**
   Die Architektur sollte eine sichere und endgültige Löschung sensibler Daten ermög-

lichen, wenn sie nicht mehr benötigt werden. Dies kann die Implementierung von Datenlöschungsrichtlinien, automatisierten Löschprozessen und Überwachungsmechanismen umfassen, um sicherzustellen, dass die Daten ordnungsgemäß gelöscht werden und nicht wiederhergestellt werden können.

Die Phase der Datenanalyse wurde nicht mit aufgenommen, dadurch, dass es sich um sensible Daten handelt, haben sie ohnehin die höchste Schutzbedarfsklasse [5].

Fazit Der Hersteller sollte ein Konzept des Datenlebenszyklus für die von ihm benötigten sensiblen Daten erstellen. Das vorgestellte Konzept ist nur ein Vorschlag und enthält die wesentlichen Punkte, die bei der Erstellung berücksichtigt werden sollten. Hierbei ist vor allem zu berücksichtigen, dass sensible Daten in den Backups auf keinen Fall unverschlüsselt vorhanden sind.

4.2.3 Prüfaspekt O.Arch_3

Die Anforderung an O.Arch_3 (vgl. Anhang A.2) ist die Definition von Richtlinien, die den Lebenszyklus und die Verwaltung des kryptographischen Schlüsselmaterials festlegen. Es werden TR-02102-2 (Technische Richtlinie für die Erzeugung von kryptographischen Schlüsseln) [63] und NIST SP 800-57 (Recommendation for Key Management) [102] genannt, die bei der Erstellung der Richtlinien berücksichtigt werden sollten. Diese Standards beziehen sich auf die Generierung, Verwaltung, Speicherung, Übertragung und Vernichtung von kryptographischen Schlüsseln.
Informationen der technischen Umsetzung können dem Prüfaspekt kryptographische Umsetzung in Abschnitt 4.5 entnommen werden.

Fazit Ähnlich wie bei O.Arch_1 wurde dieser Prüfaspekt vorgezogen und wird aber von dem Prüfaspekt Kryptografie konkretisiert. Der Inhalt der angegebenen BSI Richtlinie zur Umsetzung dieses Punktes wird dementsprechend auch in dem Abschnitt Kryptographische Umsetzung 4.5 behandelt.

4.2.4 Prüfaspekt O.Arch_6

Ziel der Anforderung O.Arch_6 (vgl. Anhang A.2) ist es, durch Authentizitäts- und Integritätsprüfung zu verhindern, dass bösartige oder manipulierte Versionen der Anwendung

installiert und ausgeführt werden. Diese Vorkehrung trägt zur Sicherstellung bei, dass die Anwendung und ihre Konfiguration vertrauenswürdig und unverändert bleiben und bietet somit einen Schutz gegen potenzielle Angriffe oder Malware.
Eine Anwendung muss über ein digitales Zertifikat verfügen, das von einer vertrauenswürdigen Zertifizierungsstelle ausgestellt wird. Dafür wird die Anwendungs-Binary (ausführbare Datei der Anwendung) digital signiert, um seine Echtheit und Integrität zu gewährleisten. Die Anwendung selbst ist dann in der Lage, während des Betriebs diese Signatur zu überprüfen, um sicherzustellen, dass das Binary nicht verändert wurde und tatsächlich von dem angegebenen Entwickler stammt. Die Authentizitäts- und Integritätsprüfung muss regelmäßig durchgeführt werden, um sicherzustellen, dass das Anwendungs-Binary nicht verändert wurde und weiterhin von einem vertrauenswürdigen Entwickler stammt.
Die Umsetzung der Authentizitäts- und Integritätsprüfung einer Anwendung mithilfe von digitalen Signaturen kann mit den folgenden Schritten umgesetzt werden. Zuerst wird ein Schlüsselpaar generiert. Das Schlüsselpaar besteht aus einem privaten Schlüssel zum Signieren und einem öffentlichen Schlüssel zum Überprüfen der Signatur. Unter Android kann dies beispielsweise mithilfe des Keytools [21] umgesetzt werden und unter iOS mithilfe des Apple Developer Program [54][49]. Danach wird die Anwendung signiert, das Zertifikat hinzugefügt und zum Schluss wird die Signatur verifiziert.

Fazit Google und Apple liefern bereits die benannten Services, um die Anforderungen umzusetzen. Es muss lediglich noch bestimmt werden, wie häufig die Anwendung eine Authentizitäts- und Integritätsprüfung durchführen soll, hierzu liefert das BSI keine Angabe. In der Regel erfolgt eine solche Prüfung bei jedem Start der Anwendung oder bei Updates der Anwendung. Aufgrund der Beschaffenheit der sensiblen Daten, die bei DiGA erhoben werden, kann es auch sinnvoll sein, regelmäßige Überprüfungen im Hintergrund durchzuführen, um sicherzustellen, dass die Anwendung während ihrer Laufzeit nicht manipuliert wurde.

4.2.5 Prüfaspekt O.Arch_7

Diese Anforderung baut auf den Prüfaspekt O.Purp_7 (siehe Abschnitt 4.1.3) auf. Denn auch in O.Arch_7 (vgl. Anhang A.2) wird betont, dass der Nutzer Informationen uber den Nutzungsumfang und die eingesetzten Sicherheitsmechanismen erhalten muss, wenn Frameworks oder Bibliotheken Dritter verwendet werden. Mögliche Sicherheitsfunktionen

hingegen wurden bereit im Abschnitt 4.2.1 benannt und werden in Bezug auf Drittanbieter in Abschnitt 4.4 genauer betrachtet.

Fazit Die Informationsweitergabe ist durch die Maßnahmen von O.Purp_7 (siehe 4.1.3) bereits abgedeckt. Ebenso die Einschränkung von nicht benötigten Funktionen. Dadurch ist auch dieser Prüfaspekt redundant.

4.2.6 Prüfaspekte O.Arch_10, O.Arch_11 und O.Arch_12

Die Prüfaspekte O.Arch_10 bis O.Arch_12 (vgl. Anhang A.2) betreffen die Sicherheitsaspekte im Zusammenhang mit Updates und der Verifizierung der Datenquelle. Sie umfassen die Überprüfung auf verfügbare sicherheitsrelevante Updates beim Start der Anwendung (O.Arch_10), die Bereitstellung des Updates über einen vertrauenswürdigen Kanal (O.Arch_11) und die Bestätigung der Authentizität der Datenquelle, wenn das Update nicht über den Apple App Store oder den Google Play Store bereitgestellt wird (O.Arch_12). Durch Erfüllung dieser Punkte wird sichergestellt, dass die Anwendung und Updates aus vertrauenswürdigen Quellen stammen und keine manipulierten oder unsicheren Versionen verwendet werden. Ergänzend darf in Bezug auf sensible Daten die Anwendung diese Art der Daten nicht mehr verarbeiten, wenn ein sicherheitsrelevantes Update zur Verfügung steht. Demnach sollte Anwendung beim Start eine Verbindung zum Server herstellen und überprüfen, ob neue sicherheitsrelevante Updates verfügbar sind. Dies kann durch den Aufruf einer speziellen API oder eines Webdienstes erfolgen, der die Informationen zu den Updates bereitstellt. Anschließend sollte der Nutzer über Pop-up Fenster oder Pushnachricht über das Update informiert werden. Gleichnamig sollten Funktionen, die sensible Daten enthalten, eingeschränkt werden.

Fazit Die Bereitstellung der Anwendung und ihrer Updates sollte vorzugsweise nur über vertrauenswürdige Kanäle wie dem Apple App Store oder dem Google Play Store bereitgestellt werden. Wenn eigene App Stores verwendet werden sollen, erhöht diese den Aufwand, da mehr Sicherheitsmaßnahmen ergriffen werden müssen. Dementsprechend ist die Empfehlung davon abzusehen. Um sicherzustellen, dass keine sensiblen Daten erhoben oder verarbeitet werden, wenn eine Aktualisierung vorliegt, ist es am einfachsten, die gesamte Anwendung zu blockieren bis der Nutzer das Update durchgeführt hat.

4.2.7 Bewertung Architektur

Bei der Bewertung wurde nicht berücksichtigt, dass sich einige Prüfaspekte um redundante Anforderungen handeln, die in anderen Abschnitten konkretisiert werden.

Bewertungsaspekt → Prüfaspekt ↓	**Aufwand**	**Umsetz-barkeit**	**Präzision der BSI Anforderung**	**Höhe der Priorität im Gesamtprojekt**
O.Arch_1, O.Arch_5, O.Arch_8, O.Arch_9	hoch	mittel	ungenau	hoch
O.Arch_2, O.Arch_4	mittel	leicht	mittel	mittel
O.Arch_3	mittel	mittel	ungenau	mittel
O.Arch_6	leicht	leicht	ungenau	mittel
O.Arch_7	hoch	schwer	mittel	hoch
O.Arch_10, O.Arch_11, O.Arch_12	leicht	leicht	eindeutig	niedrig
Gesamtbewertung	●	●	●	●

Tabelle 4.2: Bewertung Architektur

4.3 Quellcode

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Quellcode fallen. Die darin enthaltenen Prüfaspekte legen grundlegende Anforderungen für die sichere Gestaltung, Implementierung und Verwaltung des Quellcodes von DiGA fest. Dies umfasst die Prüfung und Bereinigung von Eingabedaten, den sorgfältigen Umgang mit Fehlermeldungen und Log-Dateien, die effektive Behandlung von Programmablauf-Ausnahmen, den Schutz sensibler Daten sowie die Implementierung moderner Sicherheitsmechanismen für die Quellcode-Entwicklung.

4.3.1 Prüfaspekte O.Source_1 und O.Source_2

Die Prüfaspekte O.Source_1 und O.Source_2 (vgl. Anhang A.3) erhöhen durch Eingabevalidierung und Datenbereinigung die Sicherheit des Quellcodes. Die Anforderungen

an O.Source_1 sind es, dass Eingaben aus nicht vertrauenswürdigen Quellen geprüft und validiert werden müssen, bevor sie verwendet werden. Bei nicht vertrauenswürdige Quellen handelt es sich beispielsweise um Benutzereingaben, externe Schnittstellen oder Dateiuploads. Es müssen die Sicherheitsfunktionen aus O.Arch_6 (siehe Abschnitt 4.2.4) für alle Eingaben aus nicht vertrauenswürdigen Quellen umgesetzt sein.
O.Source_2 fordert zudem, dass alle ein- und ausgehenden Daten entweder maskiert, von potenziell schadhaften Zeichen bereinigt werden sollen oder die Verarbeitung abgelehnt werden soll. Das hat zur Folge, dass potenziell schädliche oder unerwünschte Daten erkannt und neutralisiert werden können. Die Maskierung der ausgehenden Daten sieht so aus, dass insbesondere sensible Daten vor der Speicherung oder Übertragung maskiert werden sollen, um sie vor unbefugten Zugriff zu schützen. Hierfür können Verschlüsselungstechniken verwendet werden, die in dem Prüfaspekt kryptographische Umsetzung im Abschnitt 4.5 genauer betrachtet werden.

Um Angriffe wie Cross-Site Scripting (XSS) [92] oder SQL-Injection [23] zu verhindern, sollen eingehende Daten von potenziell schadhaften Zeichen bereinigt werden. Was genau unter ein schadhaftes Zeichen fällt, ist abhängig vom Kontext der Verwendung (in Datenbanken sind Hochkomma oder Prozentzeichen eventuell schädlich, bei Web/HTML Tag-Klammern). Dementsprechend muss je nach Anwendungskontext evaluiert werden, was ein schadhaftes Zeichen sein könnte. Für die Umsetzung der Eingabevalidierung würde sich eine Whitelist empfehlen. In der Whitelist kann je nach Kontext eine Liste erstellt werden mit zulässigen Werten, Zeichen oder Zeichenfolgen, die von der Anwendung akzeptiert werden dürfen. Nur die in der Whitelist gelisteten gültigen Eingaben dürfen verarbeitet werden [91]. Enthält die Eingabe nicht gültige Werte, kann sie entweder abgelehnt werden oder es wird die Methode der Escape Syntax verwendet. Durch die Verwendung einer Escape Syntax können spezielle Zeichen in einer sicheren und korrekten Weise dargestellt werden [95]. Das wird mit einem Escape-Zeichen umgesetzt, es signalisiert, dass das nachfolgende Zeichen oder die nachfolgende Zeichenfolge in einer besonderen Weise interpretiert werden soll. So wird sichergestellt, dass zum Beispiel Zeichen, die potenziell von der Anwendung ungewollte als Code interpretiert und ausgeführt werden können, als Text gelesen werden. Die Escape Zeichen sind wie die Whitelist kontextabhängig zu definieren.

Fazit Wie bereits erwähnt empfiehlt es sich eine Whitelist für die Umsetzung der Eingabevalidierung zu verwenden. Eine Empfehlung vom BSI ist, dass wenn potenziell schadhafte Zeichen erkannt worden sind, ist es zu empfehlen, diese Eingabe zu verwerfen.

Eine Bereinigung ist laut BSI zwar möglich, kann aber dazu führen, dass hier noch eine ausführliche Begründung vorgelegt werden muss, warum eine Bereinigung dem Verwerfen vorgezogen worden ist. Es sollten auf jeden Fall kontextbezogene Escape Syntaxen Anwendung finden, da die Prüfung sich auf diese fokussiert [58].

4.3.2 Prüfaspekte O.Source_3, O.Source_4 und O.Source_5

Ziel der Prüfaspekte O.Source_3 bis O.Source_5 (vgl. Anhang A.3) ist es sicherzustellen, dass Fehlermeldungen und Ausnahmen im Programmablauf angemessen behandelt werden, um sensible Daten zu schützen und die Vertraulichkeit zu wahren.
Der Quellcode muss gewährleisten, dass auftretende Exceptions abgefangen und kontrolliert behandelt werden. Zudem müssen das Auftreten und die Ursache der Exception dokumentiert werden. Folgendes ist bei dem Auftreten von Exceptions zu beachten:

- Der Stack Trace oder andere technischen Fehlerbeschreibungen dürfen auf keinen Fall dem Nutzer angezeigt werden, da sie möglicherweise Rückschlüsse auf den Quellcode und den Programmaufbau offenbaren können.
- Bei Auftreten von Exceptions, die möglicherweise mit sicherheitskritischen Auswirkungen verbunden sind, müssen gerade stattfindende Zugriffe auf sensible Daten abgebrochen werden und sensible Daten aus dem Speicher gelöscht werden.

Generell gilt, dass in Fehlermeldungen, Benachrichtigungen und Log-Dateien keine sensiblen Daten enthalten sein dürfen.

Fazit Um die Anforderungen zu erfüllen, ist es wichtig, während der Entwicklungsphase unsichere Code Passagen zu identifizieren und Mechanismen zum Abfangen von Exceptions zu implementieren - beispielsweise mittels try-catch Block, welcher eine sinnvolle Abhandlung der Exception anbietet. Dementsprechend ist es notwendig, vor und während der Entwicklung sich möglicher Fehlerquellen bewusst zu sein, diese zu dokumentieren und Mechanismen in die Implementierung einzubauen, um die Offenbarung von sensiblen Daten durch Fehler im Programmablauf zu verhindern.

4.3.3 Prüfaspekt O.Source_6

Die Anforderung O.Source_6 (vgl. Anhang A.3) bezieht sich nur auf Programmumgebungen mit manueller Speicherverwaltung. Dieser Punkt ist nur relevant, wenn man davon Gebrauch macht. Die Herausforderung bei der manuellen Speicherverwaltung besteht darin, dass unsichere Funktionen verwendet werden können, die zu Sicherheitslücken wie Buffer Overflows oder Speicherlecks führen können. Um diese Probleme zu vermeiden, werden sicherere Funktionsalternativen empfohlen. Eine gängige unsichere Funktion ist zum Beispiel *printf*. Die Größe des Ziel-Puffers, in dem die formatierte Zeichenkette geschrieben werden soll, wird von *printf* von überprüft. Wenn die formatierte Ausgabe den verfügbaren Speicherplatz überschreitet, kann es also zu einem Buffer Overflow kommen. *printf* sollte also durch die sichere Funktion *printf_s* ersetzt werden. Im Gegensatz zu *printf* erwartet *printf_s* sowohl den Formatstring als auch die maximale Anzahl von Zeichen, die in den Ziel-Puffer geschrieben werden können. Dadurch wird sichergestellt, dass die formatierte Ausgabe den verfügbaren Speicherplatz nicht überschreitet und Buffer Overflows vermieden werden [113] [69].

Fazit Da iOS (mit der Programmiersprache Swift oder Objective-C) oder Android (mit Java oder Kotlin) eine automatische Speicherverwaltung durch Garbage Collection anbieten, ist diese Anforderung in der Regel ohne weiteren Aufwand erfüllt [1]. Greift man aber doch auf die manuelle Speicherverwaltung zurück, ist es wichtig zu wissen, welche Funktionen sicher und welche unsicher sind. Dementsprechend erhöht sich der Aufwand, da für jede verwendete Funktion eingeschätzt werden muss, ob sichere Alternativen der jeweiligen Funktion existieren.

4.3.4 Prüfaspekt O.Source_7

Zur Erfüllung des Prüfaspektes O.Source_7 (vgl. Anhang A.3) sollte sich die Dokumentation der rechtmäßigen Zwecke aus 4.1 zur Hand genommen werden. Denn O.Source_7 besagt, dass alle sensiblen Daten, sobald ihr Verarbeitungszweck erfüllt worden ist, sicher gelöscht werden müssen. Dementsprechend ist die benannte Dokumentation hilfreich, um den Überblick zu behalten, welche sensiblen Daten aus welchem Zweck erhoben worden sind. So kann in der Entwicklungsphase sich darauf konzentriert werden, ob der Zweck ab einem gewissen Punkt erfüllt ist.
Interessant ist das Vorgehen in diesem Fall die Beschreibung der Testcharakteristik [58].

Darin ist beschrieben, dass geprüft wird, ob sensible Daten nach ihrer Verwendung sicher gelöscht werden. Ausnahme ist, wenn sie durch Erfüllung des Prüfaspektes O.Data_2 (siehe Abschnitt 4.7.1) geschützt sind. Bedeutet, dass wenn die Speicherung der sensiblen Daten verschlüsselt erfolgt und die genannte Anforderung hinsichtlich der Verschlüsselung erfüllt ist, müssen sie nicht gelöscht werden.
Wenn der Zweck der Verwendung erfüllt ist und die sensiblen Daten nicht mehr benötigt werden, soll ein sicheres Lösche stattfinden. Was unter Löschen versteht wird, wurde bereits in Abschnitt 4.1.2 kurz thematisiert. Das BSI empfiehlt für die sensiblen Daten eine sichere Löschung durch Datenüberschreibung im Speicher. In dem Prozess werden die Daten mehrfach mit zufälligen Bits überschrieben, um sicherzustellen, dass die ursprünglichen Daten nicht wiederhergestellt werden können. Je nach Sensibilität der Daten und den Sicherheitsanforderungen können unterschiedliche Anzahl von Überschreibungen erforderlich sein. Warum eine mehrfache Überschreibung der Daten notwendig ist, kann den folgenden Quellen [79] und [93] entnommen werden.

Fazit Die Formulierung der Testcharakteristik ist ungenau und bietet die Möglichkeit, sensible Daten unabhängig von der Erfüllung des Verwendungszweckes zu speichern, insofern die Speicherung den Anforderungen hinsichtlich der Verschlüsselung erfüllt. Nichtsdestotrotz ist es ratsam, genau zu überprüfen, ob die jeweiligen sensiblen Daten überhaupt noch für den Anwendungszweck benötigt werden und sie ab dem Zeitpunkt der Erfüllung des Zweckes sicher zu löschen. Mit dieser Art der Umsetzung ist man auf der sicheren Seite, außerdem ist es gut möglich, dass das BSI die Formulierung in einer neuen Version eventuell anpassen wird.
Wichtig ist es, bei der Löschung auch an eventuelle Kopien der Daten zu denken.

4.3.5 Prüfaspekte O.Source_8 und O.Source_9

Beide Prüfaspekte O.Source_8 und O.Source_9 (vgl. Anhang A.3) zielen darauf ab, sicherzustellen, dass die Produktiv-Version der Anwendung frei von Entwicklungs- und Debug-Funktionalitäten ist. Bei solchen Funktionalitäten handelt es sich beispielsweise um Log-Aufrufe, Entwickler-URLs, Testmethoden oder oder Debug-Mechanismen. Es muss sichergestellt werden, dass keine solcher Rückstände in der Produktiv-Version enthalten sind.

Fazit Diese Prüfaspekte verlangen, dass der Quellcode vor der Produktiv-Version aufgeräumt wird. Alle Funktionalitäten, die nicht dem Anwendungszweck, sondern der Entwicklung der Anwendung dienen, sollen also entfernt werden. Sie werden nicht für die Anwendung in der Produktiv-Phase benötigt, sondern nur in der Entwicklungsphase. Durch die Entfernung verringert man das Risiko, dass Sicherheitslücken entstehen oder Angreifer Rückschlüsse auf den Quellcode ziehen können.

4.3.6 Prüfaspekt O.Source_10

Die Anforderungen von O.Source_10 (vgl. Anhang A.3) sind abstrahiert beschrieben worden und bieten Interpretationsspielraum und mehrere Umsetzungsmöglichkeiten. Das BSI fordert darin, dass die Anwendung Sicherheitsmechanismen der entsprechenden Entwicklungsumgebung aktivieren soll. Als Beispiele werden Byte-Code-Minimierung und Stack-Protection genannt.
Bei der Byte-Code Minimierung wird der erzeugte Byte-Code der Anwendung optimiert und verkleinert. Das führt dazu, dass die potenzielle Angriffsfläche reduziert wird [22].
Stack-Protection schützt vor stackbasierten Buffer Overflows. Dazu überwacht der Mechanismus die Integrität des Stacks und erkennt unerlaubte Zugriffe auf den Speicherbereich des Stacks.
Es können und sollten weitere Sicherheitsmechanismen aktiviert werden, wie Address Space Layout Randomisation (ASLR). Bei ASLR werden Speicherbereich Adressen zufällig platziert, um es Angreifern erschweren, gezielte Angriffe basierend auf vorhersehbaren Speicheradressen durchzuführen [81].

Fazit Dieser Prüfaspekt erzeugt eine gewisse Unsicherheit, was als Minimum umgesetzt werden soll, um eine Prüfung zu bestehen. Eine genauere Auflistung, welche Sicherheitsmechanismen in welcher Entwicklungsumgebung seitens des BSI erwartet werden, wäre wünschenswert. Ein Minimum wird sicherlich die Implementierung von Byte-Code-Minimierung und Stack-Protection sein. Aber letztenendes ist es davon abhängig, welche Erwartungshaltung der Evaluator hat. Aus diesem Grund ist die Empfehlung noch weitere Sicherheitsmechanismen der jeweiligen Entwicklungsumgebung zu nutzen.

4.3.7 Bewertung Quellcode

Bewertungsaspekt → Prüfaspekt ↓	Aufwand	Umsetzbarkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
O.Source_1, O.Source_2	mittel	leicht	mittel	niedrig
O.Source_3, O.Source_4, O.Source_5	mittel	leicht	mittel	niedrig
O.Source_6	leicht	leicht	mittel	niedrig
O.Source_7	mittel	leicht	ungenau	hoch
O.Source_8, O.Source_9	leicht	leicht	eindeutig	niedrig
O.Source_10	mittel	mittel	ungenau	mittel
Gesamtbewertung	●	●	●	●

Tabelle 4.3: Bewertung Quellcode

4.4 Drittanbieter-Software

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Drittanbieter-Software fallen.
Die darin enthaltenen Prüfaspekte legen grundlegende Anforderungen für die Integration externer Software, Bibliotheken und Frameworks DiGA fest. Hierbei geht es um die sorgfältige Überwachung, Auswahl und Aktualisierung dieser Abhängigkeiten, einschließlich der regelmäßigen Überprüfung auf Schwachstellen sowie die Einhaltung von Sicherheitskonzepten. Die Prüfaspekte reichen von der Verifizierung der Quellen bis hin zur sicheren Behandlung sensibler Daten und betonen die Notwendigkeit der Vertrauenswürdigkeit und Aktualität der eingesetzten Drittanbieter-Software.

4.4.1 Prüfaspekte O.TrdP_1 und O.TrdP_6

Wie bereits in dem Fazit des Abschnitts 4.1.3 beschrieben, sollte eine Dokumentation über die Nutzung und Abhängigkeiten von externer Software, Bibliotheken und Frameworks existieren. Diese wird in dem Prüfaspekt O.TrdP_1 (vgl. Anhang A.4) gefordert.

Welche weitere Informationen mittels dieser Liste festgehalten werden sollen, darüber informiert das BSI leider nicht. Es wäre sinnvoll, dass mindestens folgende Informationen über die verwendeten Drittanbieter in der Liste enthalten sind [42]:

- Name und Versionsname des Drittanbieterprodukts / SDK / Framework / Bibliothek
- Zweck des Produkts (muss unbedingt einheitlich mit dem Prüfaspekt 4.1.3 sein)
- Hersteller mit genauer Adresse (um nachzuvollziehen ob Daten in das nicht europäische Ausland übermittelt werden)
- die übermittelten Datentypen (zu empfehlen ist es auch hier für jeden einzelnen Datentyp den Zweck der Übermittlung anzugeben)

Außerdem dürfen keine sensiblen Daten an Drittanbieter-Software weitergegeben werden. Es existiert die vom BSI definierte Ausnahme, dass Frameworks oder Bibliotheken, welche zur Verschlüsselung verwendet werden (TLS), zu diesem Zwecke sensible Daten erhalten dürfen.

Fazit O.TrdP_6 steht im Widerspruch mit O.Purp_8, denn wie in 4.1.3 bereits erläutert, dürfen sensible Daten nur mit Dritten geteilt werden, wenn dies dem primären Zweck der Anwendung dient. In diesem Prüfaspekt jedoch wird eine Weitergabe verboten. O.TrdP_6 bezieht sich nur auf Drittanbieter-Software, jedoch würde die auch unter den in O.Purp_8 benannten Begriff 'Dritte' hineinfallen. Um ganz sicherzugehen, sollte sich an die Einschränkung gehalten werden, dass sensible Daten nur an Dritte weitergegeben werden sollen, wenn zur Verschlüsselung dienen.

4.4.2 Prüfaspekte O.TrdP_2, O.TrdP_3, O.TrdP_4 und O.TrdP_8

Diese Prüfaspekte stellen sicher, dass die Anwendung auf einer aktuellen und sicheren Version externer Komponenten basiert. Eine Anforderung an diesen Prüfaspekt ist die Versionskontrolle. Das bedeutet, dass externe Bibliotheken und Frameworks nur in der Stable Version verwendet werden dürfen (O.TrdP_2, s. A.4). Das kann entweder die neuste verfügbare Stable Version sein oder vorherige Stable Version. Dadurch werden potenzielle Fehler und Schwachstellen behoben. Dass die vorherige Version verwendet werden darf, ändert nichts an der Notwendigkeit eines Versionsupdate, es dient lediglich

als zeitlichen Puffer für die Hersteller der DiGA, um das neue verfügbare Update zu prüfen und einzuspielen. Insbesondere Sicherheitsupdates müssen möglichst schnell zur Verfügung gestellt werden, hierfür muss ein entsprechendes Konzept erarbeitet werden (O.TrdP_4, s. A.4). Falls die verwendete externe Software nicht mehr gewartet wird, darf nicht sie nicht mehr verwendet werden.
Gefordert zur Erfüllung dieser Aspekte ist es, ein Sicherheitskonzept zu erstellen. Neben dem Umgang mit Sicherheitsupdates soll sich dieses Konzept mit dem Schwachstellenmanagement beschäftigen. Das heißt, Komponenten von Dritten müssen regelmäßig auf Schwachstellen überprüft werden (O.TrdP_3, s. A.4). Wenn Schwachstellen erkannt werden, muss die Weiternutzung so schnell wie möglich eingestellt werden die Höhe des Risikos der Schwachstelle definiert wie die Länge der Übergangsfrist [56].

Fazit Diese Prüfaspekte beziehen sich auf Versionskontrolle und Schwachstellenmanagement. Es handelt sich um einen fortlaufenden Vorgang, der nicht nur während der Entwicklungsphase, sondern primär in der Anwendungsphase durchgeführt werden muss. Zu erstellen und zu pflegen ist ein Konzept zur regelmäßigen Überprüfung auf (Sicherheits-) Updates und eine aktuelle Schwachstellenanalyse sowie ein Sicherheitskonzept. In der Testcharakteristik zu O.TrdP_2 prüft der Evaluator anhand der vom Hersteller erstellten Liste über eingesetzter externer Software, Frameworks und Bibliotheken, dass diese die zulässigen Stable Versionen verwenden. Das BSI referenziert sich hier auf O.Tokn_1 (Abschnitt 4.6.10), welcher allerdings über Authentifizierungstoken handelt und keine Liste von externen Komponenten anfordert. Die Vermutung liegt nahe, dass das BSI sich auf O.TrdP_1 (Abschnitt 4.4.1) beziehen wollte, in dem auch die Liste angefordert wird. Es handelt sich demnach um einen Fehler der Verlinkung.

4.4.3 Prüfaspekte O.TrdP_5 und O.TrdP_7

Die beiden Prüfaspekte betonen die Bedeutung der Überprüfung der Quelle externer Software (O.TrdP_5, s. A.4) sowie der Validierung eingehender Daten (O.TrdP_7, s. A.4).
Vor der Nutzung von Bibliotheken oder Frameworks von Drittanbietern muss geprüft werden, ob die Quelle vertrauenswürdig ist. Dies kann beispielsweise durch Überprüfung der Reputation des Anbieters, Analyse von Sicherheitsbewertungen oder Überprüfung der Verfügbarkeit von regelmäßigen Updates und Patches erfolgen [109].

Drittanbieter werden als eine nicht vertrauenswürdige Quelle angesehen. Dementsprechend muss eine Prüfung der eingehenden Daten von Drittanbieter, die bereits im Abschnitt 4.3.1 beschrieben worden ist, durchgeführt werden. Sowie entsprechende Sicherheitsfunktionen vorhanden sein, welche im Abschnitt 4.2.4 und 4.5.1 beschrieben werden.

Fazit Die Prüfaspekte sind stark an O.Source_1 (4.3.1) und O.Arch_6 (4.2.4) gekoppelt. In der Planungsphase muss die Prüfung der Drittanbieter auf Vertrauenswürdigkeit erfolgen und die Durchführung und die Ergebnisse der Prüfung dokumentiert werden. Wichtig ist vor allem zu verstehen, dass Drittanbieter trotz dieser Prüfung als nicht vertrauenswürdige Quelle angesehen werden.

4.4.4 Bewertung Drittanbieter-Software

Bewertungsaspekt → Prüfaspekt ↓	**Aufwand**	**Umsetzbarkeit**	**Präzision der BSI Anforderung**	**Höhe der Priorität im Gesamtprojekt**
O.TrdP_1, O.TrdP_6	mittel	schwer	ungenau	mittel
O.TrdP_2, O.TrdP_3, O.TrdP_4, O.TrdP_8	hoch	mittel	mittel	mittel
O.Source_5, O.Source_7	mittel	mittel	ungenau	hoch
Gesamtbewertung	🟡	🔴	🔴	🟡

Tabelle 4.4: Bewertung Drittanbieter-Software

4.5 Kryptographische Umsetzung

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Kryptographische Umsetzung fallen. Als Unterkategorie der Prüfaspekte der kryptographischen Umsetzung fallen die Prüfaspekte der Zufallszahlen, welche im Anschluss der Abhandlung der Prüfaspekte O.Cryp_1 - O.Cryp_7 als O.Rand_1 - O.Rand_4 abgehandelt werden.

Die darin enthaltenen Prüfaspekte legen grundlegende Anforderungen an die kryptographische Umsetzung für DiGA fest.

4.5.1 Prüfaspekte O.Cryp_1, O.Cryp_3, O.Cryp_4 und O.Cryp_5

O.Cryp_3 (vgl. Anhang A.5) gibt vor, dass die Wahl der kryptographischen Primitive dem aktuellen Stand der Technik entsprechen muss. Zusätzlich soll bei der Wahl der Anwendungszweck (siehe Abschnitt 4.1) berücksichtigt werden. In Bezug auf den Stand der Technik verweist das BSI für beide Prüfaspekte auf ihre Technischen Richtlinie für Kryptographische Verfahren (TR-02102-1) [64], die dort genannten Anforderungen werden im weiteren Verlauf genauer betrachtet.
O.Cryp_5 (vgl. Anhang A.5) gibt vor, dass die Stärke der kryptographischen Schlüssel dem aktuellen Stand der Technik entsprechen muss. Die Stärke des Schlüssels wird bestimmt durch die Länge des Schlüssels. In den Anforderungen an O.Cryp_3 wird für die jeweiligen Verfahren die empfohlene Mindestschlüssellänge benannt.
Folgende Punkte aus der technischen Richtlinie für Kryptographische Verfahren (TR-02102-1) [64] sollten für die Erfüllung dieser Anforderungen berücksichtigt werden:

Symmetrisches Verschlüsselungsverfahren Bei der symmetrischen Verschlüsselung wird derselbe Schlüssel sowohl für die Verschlüsselung als auch für die Entschlüsselung der Daten verwendet. Es wird also nur ein einziger geheimer Schlüssel (auch als symmetrischer Schlüssel bezeichnet) verwendet.

Blockchiffren sind eine Art von symmetrischen Verschlüsselungsverfahren, die Daten in Blöcke fester Länge verschlüsseln. Im Gegensatz zu Stromchiffren, die Daten bitweise verarbeiten, arbeiten Blockchiffren mit fest definierten Blockgrößen. Stromchiffren werden derzeit nicht vom BSI empfohlen, werden demnach auch nicht weiter behandelt und von der Nutzung abgeraten [64].

Das BSI empfiehlt aktuell folgende Blockchiffren: AES-128, AES-192 und AES-256 [64]. Der Advanced Encryption Standard (AES) ist eine weitverbreitete Blockchiffre und gilt als sicher und effizient. AES unterstützt verschiedene Schlüssellängen. Diese Schlüssellängen beziehen sich auf die Anzahl der Bits im Schlüssel, wobei AES-128, AES-192 und AES-256 entsprechend 128, 192 und 256 Bit lange Schlüssel verwenden. Die Wahl der Schlüssellänge hängt von den spezifischen Sicherheitsanforderungen einer Anwendung ab. AES-256 mit einem 256-Bit-Schlüssel gilt als besonders sicher und wird oft für

hoch sensible Daten verwendet. AES-128 und AES-192 bieten ebenfalls eine starke Verschlüsselung, jedoch mit etwas kürzeren Schlüsseln. Also je länger der Schlüssel, desto sicherer ist die Verschlüsselung. Mittels generierten Zufallszahlen (siehe Abschnitt 4.5.4) und festgelegter Schlüssellänge kann der symmetrische Schlüssel erzeugt werden [7].

Neben der Wahl der Schlüssellänge ist auch die Wahl der Betriebsart der Blockchiffren entscheidend. Die Betriebsarten bei Blockchiffren bestimmen, wie die Datenblöcke verschlüsselt und miteinander verknüpft werden. Jede Betriebsart hat spezifische Eigenschaften und kann je nach Anwendungsfall unterschiedliche Sicherheitsziele erfüllen.
Das BSI empfiehlt aktuell folgende Betriebsarten [64] [13] [103]:

- **Counter with Cipher Block Chaining Message Authentication (CCM)**
  CCM kombiniert den Counter-Modus (CTR) für die Vertraulichkeit mit dem Cipher Block Chaining-Modus (CBC) für die Authentizität (CBC-MAC). Die Datenblöcke werden verschlüsselt und verknüpft, wobei ein MAC für die Integrität der Daten berechnet wird.

- **Galois/Counter Mode (GCM)**
  GCM kombiniert den Counter-Modus (CTR) zur Verschlüsselung der Datenblöcke für Vertraulichkeit und den Galois-Modus für die Berechnung eines MAC-Tags zur Authentizität und Integrität. Im CTR-Modus werden die Datenblöcke mit einem Zähler und einem Schlüssel verschlüsselt, während im Galois-Modus ein MAC-Tag mit Hilfe einer Galois-Authentifikationsfunktion berechnet wird. Der MAC-Tag ermöglicht die Überprüfung der Authentizität und Integrität der Daten.

- **Cipher Block Chaining (CBC)**
  CBC ist eine Betriebsart, die Vertraulichkeit ermöglicht, indem sie eine Feedback-Schleife zwischen aufeinander folgenden Datenblöcken erzeugt. Jeder Klartextblock wird vor der Verschlüsselung mit dem vorherigen Chiffretextblock XOR-verknüpft. Dies führt zu einer Abhängigkeit zwischen den Datenblöcken und erhöht die Sicherheit der Verschlüsselung. Padding-Verfahren: CBC erfordert ein Padding-Verfahren, um die Klartextblöcke auf die Blockgröße des Verschlüsselungsalgorithmus aufzufüllen. Die Padding-Verfahren dienen dazu, sicherzustellen, dass alle Datenblöcke die gleiche Länge haben und den Anforderungen des Verschlüsselungsalgorithmus entsprechen. Insofern die Wahl auf die Verwendung von CBC fallen sollte, muss hinsichtlich der vom BSI empfohlenen Padding-Verfahren weiterführend recherchiert werden (Abschnitt 2.1.3 in [64]).

- **Counter Mode (CTR)**
  Der CTR-Modus ist eine Betriebsart, die Vertraulichkeit ermöglicht und eine effiziente Verarbeitung großer Datenmengen bietet. Dabei erzeugt CTR einen Strom von Pseudozufallsbits, der mit den Klartextblöcken XOR-verknüpft wird, um die Verschlüsselung durchzuführen. Der Zählerwert (Counter) wird für jeden Block inkrementiert. Dieser Modus bietet jedoch keine nativen Mechanismen zur Datenauthentifizierung oder Integritätssicherung. Daher muss in Kombination mit dem CTR-Modus ein separates Authentifizierungsverfahren verwendet werden.

Asymmetrisches Verschlüsselungsverfahren Asymmetrische Verschlüsselung, auch bekannt als Public-Key-Verschlüsselung, verwendet ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel. Der öffentliche Schlüssel wird zum Verschlüsseln von Daten verwendet, während der private Schlüssel zum Entschlüsseln der Daten verwendet wird. Das Verfahren basiert auf mathematischen Funktionen, die es ermöglichen, dass mit dem öffentlichen Schlüssel verschlüsselte Daten nur mit dem entsprechenden privaten Schlüssel entschlüsselt werden können. Dadurch bietet die asymmetrische Verschlüsselung eine sichere Kommunikation, da der private Schlüssel geheim gehalten wird und nur der Empfänger Zugriff darauf hat [75].

Das BSI empfiehlt aktuell folgende asymmetrische Verschlüsselungsverfahren:

- **Elliptic Curve Integrated Encryption Scheme (ECIES) und Discrete Logarithm Integrated Encryption Scheme (DLIES)**
  ECIES und DLIES sind eng verwandte hybride Verschlüsselungsverfahren, weshalb ihre Beschreibung in einem Punkt zusammengefasst und auch die Unterschiede deutlich gemacht werden.
  Beide Verfahren verwenden das Diffie-Hellman-Prinzip, um einen gemeinsamen geheimen Schlüssel zwischen Sender und Empfänger zu erzeugen. Der Unterschied zwischen ECIES und DLIES liegt in der Berechnung des zu erzeugenden gemeinsamen geheimen Schlüssels. Während ECIES das Diffie-Hellman-Problem in der verwendeten elliptischen Kurve nutzt, basiert DLIES auf dem Diskreten-Logarithmus-Problem (DLP) in einer geeigneten Untergruppe von F*. Der öffentliche Schlüssel des Empfängers wird mit dem erzeugten geheimen Schlüssel des Senders kombiniert, um den gemeinsamen geheimen Schlüssel zu berechnen. Dieser gemeinsame geheime Schlüssel wird dann für die symmetrische Verschlüsselung der eigentlichen Daten verwendet [75]. Das BSI legt für ECIES eine Schlüssellänge von mindestens

250 Bits fest. Für DLIES wird ab 2024 eine Schlüssellänge von 3000 Bits als Minimum gefordert, weshalb sie nun bereits umgesetzt werden sollte. Die empfohlene Mindestschlüssellänge von 3000 Bits bei DLIES ist darauf zurückzuführen, dass das DLP anfälliger für bestimmte Angriffe ist und daher längere Schlüssellängen benötigt, um einen vergleichbaren Sicherheitsgrad wie bei ECIES mit einer Schlüssellänge von 250 Bits zu erreichen [64].

- **Rivest-Shamir-Adleman (RSA)**
  RSA ermöglicht sowohl die Verschlüsselung von Nachrichten als auch die digitale Signatur von Daten. Es basiert auf dem mathematischen Problem der Faktorisierung großer Zahlen. Das bedeutet, es werden zwei große Primzahlen ausgewählt und der öffentliche Schlüssel wird aus dem Produkt dieser Primzahlen berechnet. Zum Verschlüsseln einer Nachricht wird die Nachricht in eine Zahl umgewandelt und mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Der Empfänger kann die verschlüsselte Nachricht mit seinem privaten Schlüssel entschlüsseln und die ursprüngliche Nachricht wiederherstellen [14] [85].
  Für RSA wird ab 2024 eine Schlüssellänge von 3000 Bits als Minimum vom BSI gefordert, weshalb sie nun bereits umgesetzt werden sollte [64].

Hashfunktionen Hashfunktion sind ein wichtiger Bestandteil der Kryptografie und dienen dazu, Daten in eine feste, kurze Zeichenkette (den Hashwert) umzuwandeln. Hashfunktion werden in vielen kryptographischen Anwendungen verwendet, wie zum Beispiel in der Passwortspeicherung, der Integritätsprüfung von Daten oder bei digitalen Signaturen. Es ist entscheidend, dass Hashfunktion kollisionsresistent sind, das heißt, dass es praktisch unmöglich ist, zwei unterschiedliche Eingabedaten zu finden, die denselben Hashwert erzeugen. Zusätzlich der Kollisionsresistenz muss eine Hashfunktion die sogenannte Einweg-Eigenschaft erfüllen. Das bedeutet, dass es praktisch unmöglich sein sollte, aus dem Ergebnis der Hashfunktion auf die ursprünglichen Daten rückschließen zu können. Das BSI empfiehlt in der TR-02102-1 [64] Hashfunktionen aus der SHA-2 und der SHA-3 Familie. SHA-2 verwendet den Merkle-Damgard-Konstruktion [71], bei der die Eingabe in Blöcke unterteilt und dann iterativ verarbeitet wird, während SHA-3 auf dem Sponge-Konstrukt basiert, das eine absorbierende Phase und eine Squeeze-Phase nutzt, um den Hashwert zu berechnen. Genauere Informationen der Funktionsweise und Unterschiede von SHA-2 und SHA-3 sind den Quellen [104], [24], [67] und [71] zu entnehmen.

In der TR-02102-1 [64] empfiehlt das BSI die folgenden Hashfunktionen, die nach aktuellen Stand als kryptographisch stark gelten:

- **SHA-2**
  SHA-256, SHA-512/256 (wie SHA-512, aber die Ausgabe wird auf 256 Bits abgeschnitten), SHA-384 und SHA-512
- **SHA-3**
  SHA3-256, SHA3-384 und SHA3-512

Allgemeine Anforderungen an kryptographische Verfahren Für die Verschlüsselung sollen dynamisch erzeugte Schlüssel verwendet werden. Der Prüfaspekt O.Cryp_1 (vgl. Anhang A.5) gibt vor, dass auf keinen Fall fest einprogrammierte geheime / private Schlüssel verwendet werden dürfen. Diese Anforderung dient als Schutz vor Reverse Engineering (siehe auch Abschnitt 4.11.3). Des Weiteren ist bei der Planung und Entwicklung ist zu beachten, dass kryptographische Schlüssel nur für einen Zweck eingesetzt werden dürfen (O.Cryp_4, s. A.5). Das bedeutet, dass jeder Schlüssel für eine bestimmte Aufgabe oder Funktion verwendet werden sollte und nicht für verschiedene Zwecke, um mögliche Sicherheitsprobleme zu vermeiden.

Datenauthentisierung Datenauthentisierung bezieht sich auf kryptographische Verfahren, die sicherstellen, das übermittelte oder gespeicherte Daten nicht unberechtigt verändert wurden [64]. Grundlegend ist der Ablauf so, dass der Sender einen kryptographischen Schlüssel verwendet, um eine Prüfsumme der zu authentifizierenden Daten zu berechnen, die der Empfänger dann nutzt, um die Datenintegrität und -korrektheit zu überprüfen. Die Schlüssellänge muss mindestens 128 Bits betragen. Es wird zwischen symmetrischen und asymmetrischen Verfahren unterschieden:

- **Symmetrische Verfahren**
  Sowohl Sender als auch Empfänger der Daten verwenden den gleichen kryptographischen Schlüssel. Das in der TR-02102-1 [64] vom BSI empfohlene symmetrische Verfahren zur Datenauthentisierung ist der Message Authentication Code (MAC). Ein MAC ist eine Art kryptographischer Prüfsumme, unter Verwendung eines gemeinsamen geheimen Schlüssels zwischen Sender und Empfänger berechnet wird. Der Sender verwendet den Schlüssel, um den MAC für die Nachricht zu generieren und diesen zusammen mit der Nachricht zu senden. Der Empfänger kann dann

denselben Schlüssel verwenden, um den erwarteten MAC für die empfangene Nachricht zu berechnen und diesen mit dem empfangenen MAC zu vergleichen. Wenn die beiden MACs übereinstimmen, kann der Empfänger sicher sein, dass die Nachricht unverändert und von einer vertrauenswürdigen Quelle stammt [64]. Das BSI erkennt die folgenden Verfahren als sicher an:

  - CMAC (Cipher-based Message Authentication Code) verwendet eine symmetrische Blockchiffre zur Berechnung von Authentifizierungscodes (mehr Informationen siehe [101])
  - HMAC (Hash-based Message Authentication Code) nutzt eine Hashfunktion in Kombination mit einem geheimen Schlüssel (mehr Informationen siehe [44])
  - GMAC (Galois/Counter Mode-based Message Authentication Code) kombiniert den GCM-Modus [103] mit Verschlüsselung und Authentifizierung (mehr Informationen siehe [45])

- **Asymmetrische Verfahren**
  Bei Signaturverfahren verwendet der Sender einen privaten Schlüssel, um eine digitale Signatur für eine Nachricht zu erstellen. Das bedeutet, dass die zu signierenden Daten gehasht werden und aus diesem Hashwert die Prüfsumme (Signatur) mit dem privaten Schlüssel des Senders berechnet wird. Der Empfänger kann dann die Signatur mithilfe des öffentlichen Schlüssels des Senders überprüfen, um sicherzustellen, dass die Nachricht vom angegebenen Sender stammt und während der Übertragung nicht verändert wurde. Genaue Informationen zu den vom BSI empfohlenen Signaturverfahren können der TR-02102-1 [64] Kapitel 6.3 entnommen werden.

Fazit Vor allem bei O.Cryp_3 handelt sich um einen sehr umfangreichen Prüfaspekt, der auf die Umsetzung einer anderen Technischen Richtlinie (TR-02102-1) verweist. Die wesentlichen Punkte dieses Dokumentes wurden thematisiert, welche Umsetzungsmöglichkeiten existieren und wie diese funktionieren. Es ist wichtig, darauf zu achten, dass die Umsetzung dem Anwendungsfall entspricht.
Bei der symmetrischen Verschlüsselung mit AES müssen starke Schlüssel verwendet werden. Vor allem bei sensiblen Daten sollte AES-256 verwendet werden. Hierfür sollte die Dokumentation der Datentypen aus dem Abschnitt 4.1.1 verwendet werden und dort auch die Methode der Verschlüsselung direkt ergänzt werden für die jeweiligen Datentypen. In Bezug auf die Betriebsart sind CCM oder GCM zu empfehlen. Denn zusätzlich

zur Verschlüsselung liefern beide eine kryptographisch sichere Datenauthentisierung, sodass man nicht auf separate Mechanismen zur Datenauthentisierung zurückgreifen muss. Bei dem asymmetrischen Verschlüsselungsverfahren ist es abhängig, wofür man sie einsetzen möchte. RSA wird häufig für die Verschlüsselung von kurzen Nachrichten oder zur digitalen Signatur von Daten verwendet, während DLIES und ECIES in erster Linie für die Verschlüsselung von längeren Daten oder zur sicheren Schlüsselübertragung eingesetzt werden, da sie hybride Verschlüsselungsverfahren sind und daher für größere Datenmengen besser geeignet sind. Entscheidend ist bei allen Verfahren aber die Schlüssellänge, diese sollte ausreichend groß gewählt werden.
Die Prüfaspekte O.Cryp_1, O.Cryp_4 und O.Cryp_5 knüpfen an TR-02102-1[64] an und fokussieren sich auf die Beschaffenheit der kryptographischen Schlüssel. Diese dürfen nicht fest einprogrammiert werden und nur einem einzigen Zweck dienen. Eine längere Schlüssellänge bietet eine höhere Sicherheit, da es für Angreifer schwieriger wird, den Schlüssel durch Brute-Force-Angriffe oder Faktorisierung großer Zahlen zu berechnen. Allerdings bringt eine längere Schlüssellänge auch eine größere Berechnungskomplexität und einen höheren Ressourcenbedarf mit sich, was die Leistung und Effizienz der Verschlüsselung beeinflussen kann. Daher ist es wichtig, eine ausgewogene Schlüssellänge zu wählen, die sowohl ausreichend Sicherheit bietet als auch die Anforderungen der Anwendung und der Umgebung berücksichtigt.

4.5.2 Prüfaspekt O.Cryp_2

Prüfaspekt O.Cryp_2 (vgl. Anhang A.5) legt fest, dass eine Anwendung bei der Umsetzung kryptographischer Funktionen und Protokolle auf bewährte Implementierungen zurückgreifen muss. Dabei verweist das BSI auf die Technische Richtlinie TR-02102-2 [63]. Dieser Prüfaspekt setzt die Kenntnis der kryptographischen Verfahren voraus, welche in O.Cryp_3 (siehe 4.5.1) vorgestellt worden sind.

Die Technische Richtlinie TR-02102-2 [63] bezieht sich auf die Verwendung von Transport Layer Security (TLS). TLS wird verwendet, um eine sichere Übertragung von Informationen aus der Anwendungsschicht über TCP/IP-basierte Verbindungen zu ermöglichen. TLS ist die Weiterentwicklung von Secure Socket Layer (SSL), dessen Nutzung nicht mehr empfohlen wird. Zum Verbindungsaufbau zwischen Client und Server wird ein Handshake durchgeführt. Innerhalb des Handshakes wird die TLS Version bestimmt, die kryptographischen Verfahren für die Verschlüsselung, Integritätssicherung, Schlüsseleinigung und

(ein- oder beidseitige) Authentisierung festgelegt, entschieden, welche Cipher Suites verwendet werden sollen und ein sogenanntes Master Secret ausgetauscht, aus welchem der Sitzungsschlüssel generiert wird [33].

Das BSI empfiehlt folgende TLS Versionen: TLS 1.2 und TLS 1.3. Cipher-Suiten sind eine Kombination verschiedener Algorithmen, die in einem Verschlüsselungsprotokoll verwendet werden. In TLS 1.2 werden vom BSI nur noch die Cipher-Suiten empfohlen, die Perfect Forward Secrecy (PFS) ermöglichen. PFS sorgt dafür, dass, wenn der private (langfristige) private Schlüssel des Servers kompromittiert wird, bereits stattgefundene Kommunikation sicher bleibt und folglich nicht durch den Angreifer entschlüsselt werden kann. In TLS 1.3 ist PFS standardmäßig aktiviert, da der Schlüsselaustausch nur noch auf elliptischen Kurven basiert [8] [33].
Für jeweils TLS 1.2 und TLS 1.3 existieren in der technischen Richtlinien TR-02102-2 [63] Listen mit Empfehlungen, welche Cipher-Suite, Diffie-Hellmann-Gruppen und Signaturverfahren vom BSI empfohlen werden. Definitionen der Inhalte wurden im Abschnitt 4.5.1 bereits durchgeführt. Die getroffenen Entscheidungen hinsichtlich der Krypto-Implementierungen sollten als Liste dokumentiert werden.

Fazit Bei der Wahl der Version von TLS sollte die Wahl wenn möglich auf TLS 1.3 fallen, da sie die sicherere und effizientere Version von TLS ist. Allerdings unterstützen nicht alle älteren Systeme TLS 1.3, sodass in einigen Fällen TLS 1.2 verwendet werden muss, um die Kompatibilität sicherzustellen [8] [33].
Bei jeder Entscheidung sollte berücksichtigt werden, dass das BSI angibt, bis zu welchem Jahr die dort gelisteten Empfehlungen verwendet werden dürfen (unabhängig, ob TLS 1.2 oder TLS 1.3 verwendet werden). Anhand der Versionskontrolle wird deutlich, dass diese stetig überarbeitet werden [63]. Das hat zur Folge, dass es sich empfiehlt, länger verwendbare Empfehlungen umzusetzen und einen Prozess zu definieren, um auf Aktualisierungen der Verwendungszeiträume zu achten. Diese Verwendungszeiträume sollten am besten mit in der dokumentierten Liste der Krypto-Implementierungen enthalten sein. Die Prüfaspekte Netzwerkkommunikation im Abschnitt 4.9.1 verweisen auch auf diesen Prüfaspekt.

4.5.3 Prüfaspekte O.Cryp_6 und O.Cryp_7

Beide Prüfaspekte fordern eine vor Manipulation und Offenlegung geschützte Umgebung für das Ablegen kryptographischer Schlüssel (O.Cryp_6, s. A.5) und die Durchführung kryptographischer Operationen (O.Cryp_7, s. A.5).
Folgende Beispiele für vor Manipulation und Offenlegung geschützte Umgebungen für sowohl Schlüssel als auch Operationen werden empfohlen:

- **Secure Enclave**
  Eine Secure Enclave ist ein abgesicherter Bereich innerhalb eines Prozessors, der speziell für Sicherheitsfunktionen ausgelegt ist [16]. Dieser Bereich bietet eine hohe Sicherheit und Isolierung, da sie direkt in den Prozessor integriert ist.
- **Embedded Secure Element**
  Ein Embedded Secure Element ist eine spezielle Hardwarekomponente (dedizierter Chip oder ein Sicherheitselement auf dem Hauptchip), die in ein Gerät eingebettet ist und als eigenständiger, sicherer Mikrocontroller fungiert [94].
- **Trusted Execution Environment (TEE)**
  Eine Trusted Execution Environment ist ein sicherer Bereich innerhalb eines Prozessors, der durch Hardware und Software isoliert ist von der Rich Execution Environment (REE), in der das Betriebssystem und Anwendungen laufen [37].

Fazit Es sollten möglichst nur die vom BSI empfohlenen Umgebungen zur kryptographischen Schlüsselablage und kryptographischen Operationsdurchführung verwendet werden: Secure Enclave, Embedded Secure Element (eSE) und Trusted Execution Environment (TEE) [58].

4.5.4 Prüfaspekte O.Rand_1, O.Rand_2, O.Rand_3 und O.Rand_4

Die Prüfaspekte O.Rand_1 - O.Rand_4 (vgl. Anhang A.5) formulieren folgende Anforderungen an die Beschaffenheit eines kryptographisch sicheren Zufallszahlengenerators:

- Es sollte ein starker Zufallszahlengenerator für die Erzeugung von Zufallswerten gewählt werden (O.Rand_1) .
- Der Zufallszahlengenerator sollte eine hohe Entropie haben (O.Rand_2).

- Und es sollte sich um einen Hardware Zufallszahlengenerator handeln, der keine Startwerte erlaubt. Für den Fall, dass die Plattform keinen zur Verfügung stellt, soll dem Zufallszahlengenerator ein Startwert, der sogenannte Seed, zugewiesen werden (O.Rand_3). Für den Seed gelten folgende Anforderungen: Er soll sich aus mindestens drei voneinander unabhängigen Systemparametern zusammensetzen. Ein Beispiel wäre die aktuelle Systemzeit, Prozess-ID und Systemzähler. Bei Erstellung eines Seeds soll ein geeigneter Zufall aus einer geeigneten externen Quelle bezogen werden, in mobilen Geräten können dies zum Beispiel Bewegungssensoren sein (O.Rand_4).

Das BSI weist in der TR-02102 [64] darauf hin, dass ungeeignete Zufallszahlengeneratoren starke kryptographische Mechanismen entscheidend schwächen können und aus diesem Grund bei kryptographischen Anwendungen besonders darauf geachtet werden muss, dass geeignete Zufallszahlengeneratoren eingesetzt werden.

Das BSI spezifiziert in der technischen Richtlinie TR-03116 [62] Zufallszahlengeneratoren, welche die Anforderungen an eine hohe Entropie und Stärke erfüllen und teil diese Zufallszahlengeneratoren in Funktionsklassen ein [62] [110]:

1. **Deterministischen Zufallszahlengeneratoren**

   - DRG.3 - basiert auf kryptographischen Algorithmen. Zur Umsetzung von DRG.3 wird ein kryptographisch starker Algorithmus verwendet, der auf einem geheimen Initialwert (Seed) basiert. Dieser Seed wird durch geeignete Quellen bereitgestellt (siehe oben). Mit Hilfe des Algorithmus werden dann die Zufallszahlen deterministisch erzeugt.

   - DRG.4 - basiert auf nicht-kryptographischen Algorithmen. DRG.4 verwendet keine kryptographischen Algorithmen, sondern setzt auf mathematische Verfahren zur Generierung von Zufallswerten. Es ist wichtig, dass diese Verfahren ausreichend sicher sind, um eine ausreichende Qualität der Zufallszahlen zu gewährleisten.

2. **Physikalischen Zufallszahlengeneratoren**

   - PTG.3 - basiert auf physikalischen Prozessen oder Naturphänomenen. PTG.3 nutzt physikalische Effekte, wie beispielsweise atmosphärische Rauschen, thermisches Rauschen oder Quanteneffekte, um echte Zufallszahlen zu erzeugen.

Diese sind nicht deterministisch und bieten eine hohe Qualität und Sicherheit der Zufallswerte.

3. **Nicht-physikalische nicht-deterministische Zufallszahlengeneratoren**

   - NTG.1 - basiert auf nicht-physikalischen Prozessen und ist nicht-deterministisch. NTG.1 nutzt beispielsweise Betriebssystem-Entropiequellen, Benutzereingaben oder andere externe Quellen, die als nicht-deterministisch gelten, um Zufallswerte zu erzeugen. Die Qualität und Sicherheit dieser Zufallswerte hängt von der Entropie der Quellen ab und kann je nach Implementierung variieren.

Fazit Es ist zu beachten, dass deterministische Zufallszahlengeneratoren in der Regel nicht die gleiche Qualität und Sicherheit bieten wie physikalische Zufallszahlengeneratoren, die auf physischen Prozessen oder Naturphänomenen basieren. Daher ist in sicherheitskritischen Anwendungen ist der Einsatz von echten Zufallszahlengeneratoren empfehlenswert.

4.5.5 Bewertung Kryptographische Umsetzung und Zufallszahlen

In die Bewertung fließen hinsichtlich der Präzision des BSI nur die Formulierungen der Prüfaspekte mit ein. Der Verweis auf weitere Richtlinien, in denen lange nach Empfehlungen des BSI gesucht werden muss, werden als ungenau bewertet.

Bewertungsaspekt → Prüfaspekt ↓	**Aufwand**	**Umsetz-barkeit**	**Präzision der BSI Anforderung**	**Höhe der Priorität im Gesamtprojekt**
O.Cryp_1, O.Cryp_3, O.Cryp_4, O.Cryp_5	hoch	schwer	ungenau	hoch
O.Cryp_2	hoch	mittel	mittel	hoch
O.Cryp_6, O.Cryp_7	hoch	mittel	mittel	hoch
O.Rand_1, O.Rand_2, O.Rand_3, O.Rand_4	mittel	mittel	ungenau	hoch
Gesamtbewertung				

Tabelle 4.5: Bewertung Kryptographische Umsetzung und Zufallszahlen

4.6 Authentifizierung

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Authentifizierung fallen. Als Unterkategorien der Authentifizierung Prüfaspekte fallen die Prüfaspekte: Authentifizierung über Passwort (O.Pass_1 bis O.Pass_5), Authentifizierung über Biometrie (O.Biom_1 bis O.Biom_8), zustandsbezogene Authentifizierungsmaßnahmen (O.Sess_1 bis O.Sess_6) und zustandslose Authentifizierungsmaßnahmen (O.Tokn_1 bis O.Tokn_6). Diese werden im Anschluss der Abhandlung der Prüfaspekte O.Auth_1 bis O.Auth_12 behandelt. Die darin enthaltenen Prüfaspekte legen grundlegende Anforderungen an die Authentifizierung für DiGA fest.

4.6.1 Prüfaspekte O.Auth_1, O.Auth_2, O.Auth_3 und O.Auth_4

Der Prüfaspekt O.Auth_1 (vgl. Anhang A.6) fordert die Erstellung eines Konzepts, welches die Authentifizierung, die Autorisierung und das Beenden einer Anwendungssitzung festlegt und dokumentiert. Die Anforderungen an die Umsetzung der Authentifizierung werden durch die Prüfaspekte O.Auth_3 und O.Auth_4 spezifiziert (vgl. Anhang A.6).

Authentifizierungsmechanismen und Autorisierungsfunktionen sollen außerdem getrennt realisiert werden (O.Auth_2, s. A.6).

Zunächst ist es sinnvoll, die Begriffe der Authentisierung, Authentifizierung und der Autorisierung zu klären. Ein Nutzer, der die Anwendung benutzen will, muss sich zunächst authentisieren. Authentisierung ist die Erbringung eines Identitätsnachweises, welcher von der Anwendung in dem Prozess der Authentifizierung geprüft werden soll. Authentifizierung ist folglich der Prozess der Überprüfung der Echtheit der Identität des Nutzers. Nach erfolgreicher Authentifizierung, also der Bestätigung der Identität, wird dem Nutzer Zugang zu den Funktionen der Anwendungen gewährt, für die er autorisiert ist. Der Nutzer hat also Zugriffsrechte, die anhand seiner Rolle innerhalb der Anwendung erteilt werden [12] [47].

Es folgen Möglichkeiten, die umgesetzt werden können und sollten, um die genannten Anforderungen zu erfüllen:

1. **Zwei-Faktor-Authentisierung (basierend auf den Empfehlungen des BSI [61])**
   Ein Passwort allein bietet nach aktuellen Sicherheitsstandards nicht mehr den sichersten Schutz. Es ist daher zwingend notwendig, dass die Anwendung die Kombination oder das Aufeinanderfolgen von zwei Identifikationsfaktoren verlangt, um die Authentifizierung durchzuführen. Hierbei stehen die Kategorien Wissen, Besitz und Inhärenz im Fokus. Die gewählten Faktoren müssen unterschiedlichen Kategorien angehören. Die Kategorie Wissen basiert auf Passworteingabe (in Kombination mit einer weiteren Kennung wie Name, E-Mail oder ID). Für gewöhnlich ist das der erste Schritt der Zwei-Faktor-Authentisierung. Konkrete Anforderungen an Passwörter werden in dem Abschnitt 4.6.5 behandelt. Unter die Kategorie Besitz fällt ein weiteres Gerät wie ein bereits registriertes Smartphone, auf das der Nutzer eine Einmalkennung wie beispielsweise eine TAN übermittelt bekommt, welche als zweiten Authentisierungsschritt in der Anwendung eingegeben werden muss. Die Kategorie Biometrie setzt ein Vorhandensein von zuvor erfassten biometrischen Merkmalen voraus, welches zur Überprüfung der Identität verwendet werden soll. Unter einzigartige biometrische Merkmale fallen Fingerabdruck, Gesicht und Retina. Für die Biometrische Überprüfung ist eine Lebenderkennung notwendig. Weitere Anforderungen der Biometrie können den Abschnitten 4.6.6, 4.6.7 und 4.6.8 entnommen werden.

2. **Rollenkonzept**
   Die Anwendung sollte nach Bedarf verschiedene Rollen definieren, die unterschiedliche Zugriffsrechte haben. Nutzer können dann entsprechend ihrer Rolle nur auf bestimmte Funktionen oder Daten zugreifen. Welcher Nutzer welche Rolle und welche Rolle Zugriff auf welche Daten hat, muss hierbei genau dokumentiert werden. Auch in der Anwendungsphase muss das Rollenkonzept aktuell gehalten werden [105].

3. **Session-Management**
   Die Anwendung sollte eine Möglichkeit zum kontrollierten Beenden einer Anwendungssitzung anbieten. Das sichere Beenden einer Anwendungssitzung sollte zum Beispiel durch automatisches Ausloggen nach einer bestimmten Zeit der Inaktivität erfolgen (siehe O.Auth_8 in Abschnitt 4.6.3) oder auch wenn die Anwendung in den Hintergrundbetrieb versetzt wird (siehe O.Auth_7 in Abschnitt 4.6.3). Weiterführend sollte darauf geachtet werden, dass nur eine aktive Anwendungssitzung erlaubt ist.

4. **Separate Realisierung von Authentifizierungsmechanismen und Autorisierungsfunktionen**
   Die Anwendung kann verschiedene Module oder Komponenten verwenden, um die Authentifizierung und die Autorisierung zu implementieren. Diese Module können unabhängig voneinander arbeiten und haben klare Schnittstellen für die Kommunikation (Modularität).

5. **Bewertung eines Authentifizierungsvorgangs**
   Neben der Authentifizierung der Zwei-Faktor-Authentisierung sollen auch zusätzlich Informationen in die Bewertung der Authentifizierung mit einbezogen werden. Das können kontextbasierte Informationen sein, um den Kontext des Zugriffs zu bewerten, wie beispielsweise Geräteinformationen (Geräte-IDs, Betriebssystemversion und Sicherheitsfunktionen), IP-Adresse, GPS-Standort (Erkennung ungewöhnlicher Standorte) und Zeit des Zugriffs. Auch sollten die Informationen über den verwendeten WiFi-Zugangsknoten geprüft werden, dieser kann zur Verifizierung der Netzwerkverbindung dienen und verdächtige Zugriffe von ungewöhnlichen Standorten erkennen. Die Bewertung soll mögliche Anomalien oder verdächtige Aktivitäten zu erkennen und geeignete Maßnahmen je nach Risiko ergreifen, wie in jeden Fall den Nutzer entsprechend in Kenntnis zu setzen (siehe O.Auth_5 4.6.2).

Fazit Bezüglich der Zwei-Faktor-Authentisierung wird empfohlen, dass insofern der Faktor Besitz gewählt wird, der Einmalcode nicht auf das Gerät gesendet werden soll, an dem die Authentisierung stattfindet. Es werden also zwei verschiedene Geräte empfohlen: Eines, das zur Verwendung der Anwendung dient und ein anderes, an dem der zweite Verifikationsschritt durchgeführt wird. Abhängig vom Anwendungszweck kann es aber ein Use Case sein, dass der Nutzer die Anwendung unterwegs ohne das Vorhandenseins eines zweiten Gerätes nutzen möchte. Ein weiterer Use Case ist es, dass möglicherweise nicht jeder Nutzer den Zugang oder im Besitz eines zweiten Gerätes ist. Hinsichtlich der Biometrischen Erkennung kann es zu dem Fall kommen, dass das Gerät nicht über entsprechende Funktionen verfügt. Bei der Wahl des zweiten Faktors müssen in der Planung folglich zuerst die Use Cases und die Nutzergruppen (Zielgruppe) der Anwendung spezifiziert werden, bevor eine Aussage getroffen werden kann, welcher zweiter Faktor sich am besten eignet. Die Umsetzung der Authentifizierung stellt gerade für DiGA ein großes Hindernis dar. Es sollte jedem Nutzer barrierefrei der Zugang zu DiGA zur Verfügung gestellt werden. Allerdings kann es schnell der Fall sein, dass bestimmte Zielgruppen nur über ein Gerät verfügen, dass die benötigten technischen Voraussetzungen (Gesichtserkennung, Fingerabdrucksensor) nicht erfüllt oder nicht über zwei Geräte verfügen oder mit der Koordination von zwei Geräten nicht umgehen können oder sich den Bestätigungscode und Passwort nicht merken können. All diese Szenarien müssen bei der Planung berücksichtigt und nach Relevanz und Wahrscheinlichkeit bewertet werden, denn eine zufriedenstellende Lösung hinsichtlich der Benutzerfreundlichkeit wird aufgrund der derzeitigen Anforderungen nicht möglich sein.

4.6.2 Prüfaspekte O.Auth_5 und O.Auth_6

Diese beiden Prüfaspekte fokussieren sich auf Anforderungen hinsichtlich ungewöhnlicher Anmeldeversuche. Der Nutzer soll über ungewöhnliche Anmeldeversuche informiert werden (O.Auth_5, s. A.6). Hierfür dem Nutzer Benachrichtigungen per E-Mail, SMS oder Pushnachrichten senden, wenn ungewöhnliche Anmeldevorgänge festgestellt werden, wenn zum Beispiel von einem unbekannten Gerät oder aus einem unbekannten Standort auf das Benutzerkonto zugegriffen wird (siehe O.Auth_4 in Abschnitt 4.6.1). Alternativ oder zusätzlich kann eine Zugriffshistorie dem Nutzer leicht zugänglich zur Verfügung gestellt werden, damit er über alle Anmeldungen und Aktivitäten informiert ist.

Um die Anforderungen von O.Auth_6 (vgl. Anhang A.6) hinsichtlich der Umsetzung der Maßnahmen zur Verhinderung von Ausprobieren von Login-Parametern (wie Passwörter)

umzusetzen, können zeitliche Verzögerungen zwischen Anmeldeversuchen eingeführt werden. Um Brute-Force-Angriffe zu erschweren, kann die Anwendung die Verwendung von Captchas verlangen, um zu bestätigen, dass ein echter Nutzer versucht, sich anzumelden und nicht ein automatisierter Angreifer. Die Anwendung kann auch eine Sperrfunktion einführen, die das Konto nach einer bestimmten Anzahl von fehlgeschlagenen Anmeldeversuchen vorübergehend sperrt oder deaktiviert, wobei die letztere Umsetzungsmöglichkeit nicht in der Aufzählung des BSI benannt worden ist [58].

Fazit Der Nutzer erhält Informationen über potenziell verdächtige Anmeldevorgänge und wird vor unerlaubtem Zugriff geschützt, indem das Ausprobieren von Login-Parametern erschwert oder verhindert wird. Es wird vor allem geprüft, dass dem Nutzer leicht zugänglich die Möglichkeit gegeben wird, Informationen zu Anmeldevorgängen nachzuvollziehen. Das Erschweren und Verhindern des Ausprobierens von Login-Parametern ist zwar eine Anforderung, die erfüllt werden muss, dennoch birgt sie keine großen Sicherheitslücken, da aufgrund der Zwei-Faktor-Authentisierung das Brute-forcen des einen Faktor noch keinen Zugriff auf die Anwendung ermöglicht.

4.6.3 Prüfaspekte O.Auth_7, O.Auth_8, O.Auth_9, O.Auth_10 und O.Auth_12

Die Anwendung muss eine erneute Authentifizierung nach O.Auth_3 (siehe Abschnitt 4.6.1) bei folgenden Fällen fordern:

- Nach angemessener Zeit (grace period) bei unterbrochener Anwendung (Wechsel in den Hintergrundbetrieb) (O.Auth_7, s. A.6)
- Nach angemessener Zeit (idle time) in der sie nicht aktiv verwendet wurde (O.Auth_8, s. A.6)
- Nach angemessener Zeit (active time) in der sie dauerhaft aktiv verwendet wurde (O.Auth_9, s. A.6)
- Bei Nutzeränderungen der Authentisierungsdaten. Eine erneute Authentifizierung ist notwendig, bevor die Authentisierungsdaten geändert werden dürfen. Dies gilt auch bei Zurücksetzen des Passwortes. Für das Zurücksetzen des Passwortes muss

ein geeigneter Ablauf geplant werden. Das BSI legt fest, dass bei beispielsweise Sicherheitsabfragen darf die Antwort nicht leicht zu erraten oder gar aus möglicherweise öffentlichen Informationen ermittelbar sein (wie Mädchenname der Mutter) (O.Auth_10, s. A.6) [58].

Wie bereits erwähnt, soll sich die erneute Authentifizierung an die Anforderungen von O.Auth_3 (siehe 4.6.1) richten, das bedeutet eine Zwei-Faktor-Authentisierung ist notwendig.
Das BSI offeriert aber die Möglichkeit in der Anwendungssitzung, also der Zeitraum, in dem der Benutzer bereits angemeldet ist, dass der zweite Faktor vom Hintergrundsystem erzeugt werden kann. Eine solche Umsetzung erleichtert den Nutzer die erneute Anmeldung durch die erwähnten Timeouts nach grace period, idle und ative time.
Das BSI empfiehlt für die Umsetzung zum Beispiel Time-based One-Time Password (TOTP). Bei TOTP wird der zweite Faktor in Form eines zeitbasierten Einmalpasswortes erzeugt, das alle 30 Sekunden ändert. Der Benutzer erhält dieses Einmalpasswort auf seinem Mobilgerät über eine Authentifizierungsanwendung wie beispielsweise Google Authenticator [106]. Der genaue (mathematische) Aufbau des TOTP Algorithmus kann dem RFC6238 von der Internet Engineering Task Force (IETF) [86] entnommen werden.

Fazit Zu der Dauer der grace period, idle und active time gibt das BSI keine Aussagen, welche Dauer als angemessen angesehen wird. Tendenziell sollten grace period und idle time möglichst kurz gewählt werden (beispielsweise zwischen 1-5 Minuten), während die active time etwas länger, je nach Art der Anwendung, gewählt werden sollte (beispielsweise zwischen 15-60 Minuten). Die Entscheidung sollte jedoch immer unter Berücksichtigung der Benutzerfreundlichkeit getroffen werden. Wenn die Zeitspannen zu kurz sind, könnte dies zu häufigen und störenden Authentifizierungsanforderungen führen und die Benutzererfahrung beeinträchtigen.
In Bezug auf die Erzeugung des zweiten Faktors durch das Hintergrundsystem bei der Authentifizierung in der Anwendungssitzung sollte die Möglichkeit nur für O.Auth_7 - O.Auth_9 bestehen. Für O.Auth_10 sollten die zwei Faktoren verwendet werden, die auch für die Anmeldung gefordert sind, da Änderungen der Authentisierungsdaten ein höheres Sicherheitsrisiko darstellen.

4.6.4 Prüfaspekt O.Auth_11

O.Auth_11 (vgl. Anhang A.6) fordert, dass neben der Nutzer-Authentifizierung, muss auch eine Authentifizierung an der Schnittstelle zwischen Anwendung und Hintergrundsystem von der Anwendung unterstützt werden muss. Die Anwendung sollte eine sichere Schnittstelle implementieren, um mit dem Hintergrundsystem zu kommunizieren. Für die Authentifizierung zwischen der Anwendung und dem Hintergrundsystem können verschiedene Methoden verwendet werden.
Beispielsweise werden API-Schlüssel und OAuth-Token häufig verwendet, um sich gegenüber einem Hintergrundsystem (Server) zu authentifizieren:

- **API-Key**
  Ein API-Key ist ein geheimer Schlüssel, der der mobilen Anwendung zugewiesen wird. Die Anwendung sendet den API-Schlüssel bei jeder Anfrage an das Hintergrundsystem mit, um ihre Identität zu bestätigen und Zugriff zu erhalten. Der API-Schlüssel wird normalerweise als Parameter in der Anfrage oder als Teil des HTTP-Headers übertragen [77].

- **OAuth-Token**
  OAuth ist ein Protokoll, das in mobilen Anwendungen häufig für die Authentifizierung und Autorisierung von Benutzern verwendet wird. Statt eines statischen Schlüssels wird ein temporäres Zugriffstoken (OAuth-Token) verwendet, das der Anwendung zugewiesen wird, nachdem der Benutzer die Anwendung autorisiert hat. Das Token wird in jeder Anfrage an das Hintergrundsystem übertragen, um die Identität des Benutzers zu bestätigen und Zugriff auf geschützte Ressourcen zu erhalten [19] [96].

Fazit Der Hauptunterschied zwischen API-Schlüsseln und OAuth-Token liegt in der Art der Authentifizierung. API-Schlüssel sind statische, geheime Schlüssel, die einer Anwendung zugeordnet sind und keinen weiteren Interaktionsprozess erfordern. OAuth-Token hingegen sind temporäre Zugriffstoken, die nach der Autorisierung des Benutzers ausgestellt werden und es der Anwendung ermöglichen, in seinem Namen auf geschützte Ressourcen zuzugreifen. Das BSI gibt keine konkrete Art der Authentifizierung an und geprüft wird laut Testcharakteristik auch nur das Vorhandensein einer Authentifizierung mit den Hintergrundsystem. Durch das Fehlen der Angaben hinsichtlich der Umsetzung

sollte die Wahl genau dokumentiert und begründet werden und bei der Wahl darauf geachtet werden, dass Anforderungen an die Sicherheit erfüllt sind.

4.6.5 Prüfaspekte O.Pass_1, O.Pass_2, O.Pass_3, O.Pass_4 und O.Pass_5

Die Anforderungen an diese Prüfaspekte beziehen sich auf die Authentifizierung mittels Benutzername und Passwort. O.Pass_1 und O.Pass_2 (vgl. Anhang A.6) beziehen sich auf die Stärke des Passwortes. Es müssen starke Passwortrichtlinien existieren. Das BSI stellt in ihren Sicherheitsempfehlungen 'Sichere Passwörter erstellen' [57] zwei mögliche Strategien zur Erstellung eines sicheren Passwortes vor:

- Langes und weniger komplexes Passwort: Eine Länge von mindestens 25 Zeichen. Es müssen mindestens zwei Zeichenarten verwendet werden.
- Kürzeres und komplexeres Passwort: Eine Länge von mindesten acht Zeichen. Es müssen mindestens vier Zeichenarten verwendet werden.

Als Zeichenarten gelten: Alle verfügbaren Zeichen wie Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen (Leerzeichen, '?', '!', '+' etc.) [57]. Bei der Passwortauswahl können dem Nutzer Informationen über die Stärke des gewählten Passwortes angezeigt werden, damit der Nutzer eine weitere Leitlinie hat um sein Passwort noch sicherer zu machen. Bei der Entwicklung muss allerdings darauf geachtet werden, dass diese Informationen über die Stärke des gewählten Passwortes weder im Anwendungsspeicher, noch im Hintergrundsystem persistiert werden.

O.Pass_3 und O.Pass_4 (vgl. Anhang A.6) beziehen sich darauf, dass der Nutzer die Möglichkeit haben soll sein Passwort ändern zu können. Änderungen sowie das Zurücksetzen des Passwortes müssen in einem Protokoll festgehalten werden und der Nutzer soll über diese Aktivität informiert werden. In dem Abschnitt 4.6.2 wurde bereits eine solche Informationsweitergabe (in Hinblick auf ungewöhnliches Verhalten) beschrieben. Diese Anforderungen können gemeinsam mit einem solchen Protokoll umgesetzt werden. Dass die Änderung und das Zurücksetzen des Passwortes Authentifizierungsmaßnahmen benötigen, wurde bereits in Abschnitt 4.6.3 beschrieben, diese müssen auf jeden Fall berücksichtigt werden.

Die Anforderung O.Pass_5 (vgl. Anhang A.6) legt fest, dass Passwörter, die gespeichert werden gehasht und mit einem Salt versehen werden müssen. Hash-Funktionen sorgen

dafür, dass das Passwort nicht im Klartext gespeichert wird, sondern als Hash-Wert, der nicht zurück in das ursprüngliche Passwort umgewandelt werden kann. Die entsprechenden Anforderungen an die Hashfunktion kann dem Abschnitt 4.5.1 entnommen werden. Salts werden verwendet, um den Hash-Wert jedes Passworts einzigartig zu machen und vor Rainbow-Angriffen zu schützen. Ein Salt ist eine zufällig gewählte Zeichenfolge, die an einen ursprünglichen Text angehängt wird, bevor dieser weiterverarbeitet wird [64]. Also wird beispielsweise der Salt an ein Passwort angehängt, bevor dieses in die Hashfunktion eingegeben wird.

Fazit Das Minimum der vom BSI empfohlenen Passwort-Richtlinien sollte erfüllt werden. Es wäre, aufgrund des sensiblen Kontextes besser, noch mehr als die empfohlene Anzahl an Zeichenarten zu verwenden. Bei Speicherung des Passwortes sollte das Passwort vorzugsweise mindestens mit SHA256 gehasht und mit einem Salt versehen werden.

4.6.6 Prüfaspekte O.Biom_1, O.Biom_2, O.Biom_3 und O.Biom_4

Diese und die folgenden Prüfaspekte an die Biometrie definieren Anforderungen an die Verwendung von biometrischen Systemen zur Authentifizierung. Generell, dürfen biometrische Systeme nicht als alleiniger Authentifizierungsmechanismus eingesetzt werden, sondern nur als Teil der Zwei-Faktor-Authentisierung (O.Biom_1, s. A.6).
Für O.Biom_2 (vgl. Anhang A.6) definiert das BSI Mindestanforderungen an biometrische Systeme hinsichtlich Qualität und Eigenschaften. Dafür wird auf den Anhang C der TR-03161 [58] verwiesen. Die Anforderungen beziehen sich auf das Betriebssystem (Mindestanforderung: Android Version 9, iOS Version 15), die Performance und die Erkennung von Präsentationsangriffen (Presentation Attack Detection). Detaillierte Informationen bezüglich Presentation Attack Detection (PAD) sind den Quellen [43] und [73] zu entnehmen. Maßgeblich ist die Falschakzeptanzrate (FAR), welche die Wahrscheinlichkeit angibt, dass das System einen unberechtigten Benutzer fälschlicherweise akzeptiert und die Falschrückweisungsrate (FRR), welches die Rate ist, mit der das System einen legitimen Benutzer fälschlicherweise ablehnt. Das biometrische System muss eine FAR von 1 zu 33.333 oder besser aufweisen. Die FAR muss unter den Bedingungen eines 'validen Angriffs' gemäß der Definition in der Norm ISO/IEC 30107-1 [100] erreicht werden. Das bedeutet, dass das System auch unter simulierten Betrugsversuchen, bei denen keinerlei Aufwand unternommen wird, um das System zu täuschen, zuverlässig funktionieren muss. Der Hersteller des Smartphones, auf dem das biometrische System läuft, muss

die FRR für jede verwendete biometrische Modalität (z. B. Fingerabdruck, Gesichtserkennung) angeben und dokumentieren. Das biometrische System muss eine Presentation Attack Detection (PAD) implementieren. Das bedeutet, dass es in der Lage sein muss, Angriffe zu erkennen, bei denen ein Angreifer versucht, das System mit gefälschten biometrischen Merkmalen zu täuschen. Das biometrische System muss gegen verschiedene vordefinierte Angriffstypen getestet werden, die als Presentation Attack Instrument (PAI) Species bezeichnet werden. Das BSI stellt eine Liste solcher Angriffstypen zur Verfügung, die während der PAD-Prüfung getestet werden müssen. Der technischen Richtlinie TR-03166 [99] sind weitere Anforderungen und Informationen zu biometrischen Systemen zu entnehmen.

Die Anwendung muss vor jeder Authentifizierung sicherstellen, dass die Hardware den in O.Biom_2 definierten Anforderungen entspricht (O.Biom_3, s. A.6). Sollten die Anforderungen von O.Biom_2 nicht mittels Hardware erfüllt werden können, hat der Hersteller der DiGA die Möglichkeit die Anforderungen softwareseitig zu erfüllen (O.Biom_4, s. A.6).

Fazit Da die Verwendung biometrischer Systeme nicht zwingend notwendig ist, da sie nicht als alleinstehender Authentisierungsfaktor zulässig ist und die Wahl für den zweiten Authentisierungsfaktor auch auf die Kategorie Besitz fallen kann (siehe 4.6.1), wurden nur die Grundanforderungen erläutert. In dem bereits vorgestellten Dokument [99] werden je nach Sicherheitsstufe weitere Anforderungen definiert. Dieses Dokument sollte verwendet werden, wenn die DiGA als zweiten Authentisierungsfaktor biometrische Systeme verwenden soll.

4.6.7 Prüfaspekte O.Biom_5 und O.Biom_6

Dem System muss mindestens eine biometrische Referenz vorliegen. Das muss vor der Anwendung des biometrischen Systems sichergestellt werden. Diese biometrische Referenz dient als Vergleichswert (O.Biom_5, s. A.6). Die Anwendung muss feststellen, wenn die biometrische Referenz ohne Autorisierung verändert worden ist. Die Anmeldung muss dann abgelehnt werden (O.Biom_6, s. A.6).

Fazit Zusammenfassend stellen diese beiden Prüfaspekte sicher, dass die biometrischen Referenzen korrekt erfasst und unverändert bleiben, um eine zuverlässige und sichere biometrische Authentifizierung in der mobilen Anwendung zu gewährleisten. Sie erweitern den bereits beschriebenen Aspekt der Änderung der Authentisierungsdaten ohne ausreichende Authentifizierung, den O.Auth_10 beschreibt (siehe Abschnitt 4.6.3).

4.6.8 Prüfaspekte O.Biom_7 und O.Biom_8

O.Biom_7 (vgl. Anhang A.6) stellt die Anforderung, dass die Auswertung der biometrischen Authentifizierung durch betriebssystemeigene Funktionalitäten erfolgen soll.
Unter Android kann die BiometricPrompt API verwendet werden, um die Auswertung des Authentifizierungsergebnisses durch das Betriebssystem durchzuführen. Die genaue Umsetzung kann der Dokumentation [28] entnommen werden.
Unter iOS kann das Local Authentication Framework verwendet werden, um auf die betriebssystemeigenen Funktionalitäten für die biometrische Authentifizierung zuzugreifen. Die genaue Umsetzung kann der Dokumentation [52] entnommen werden.

O.Biom_8 (vgl. Anhang A.6) bezieht auch auf den Schwellwert fur biometrische Authentifizierungsversuche. Die Anwendung muss die Anzahl der fehlgeschlagenen biometrischen Authentifizierungsversuche zählen und nach Erreichen der festgelegten Grenze die biometrische Authentifizierung temporär deaktivieren. Der Benutzer kann dann nur noch alternative Authentisierungsmethoden verwenden, bis er sich erfolgreich mit einem anderen Faktor (z. B. Passwort oder PIN) authentifiziert hat. Nach erfolgreicher alternativer Authentifizierung kann die biometrische Authentifizierung wieder aktiviert werden.

Fazit Vor allem in Bezug auf Prüfaspekt O.Biom_8 wird deutlich, dass die Implementierung von einen biometrischen System als alleinigen zweiten Faktor für die Zwei-Faktor-Authentisierung nicht ausreichend ist. Es müssen demnach auch von den anderen beiden Authentisierungskategorien (Wissen und Besitz) Authentifizierungsverfahren eingerichtet sein. Da es sich um sensible Daten handelt, ist ein geeigneter Schwellwert für die maximale Anzahl von aufeinanderfolgenden fehlgeschlagenen biometrischen Authentifizierungsversuchen, sehr niedrig zu wählen. Das BSI legt keine genaue Anzahl der Versuche fest. Ein typischer Schwellwert liegt oft zwischen drei und fünf fehlgeschlagenen Versuchen, bevor die biometrische Authentifizierung temporär deaktiviert wird, wobei vorzugsweise drei

gewählt werden sollte. Gerade bei beispielsweise Gesichtserkennung kann dieser Schwellwert schnell erreicht werden und eine Entsperrung ist danach notwendig durch andere Authentisierungsmethoden.

4.6.9 Prüfaspekte O.Sess_1, O.Sess_2, O.Sess_3, O.Sess_4, O.Sess_5 und O.Sess_6

Diese Prüfaspekte definieren die Anforderungen an das Session-Handling als zustandsbezogene Authentifizierungsmaßnahme. Das Session-Handling bezieht sich auf die Verwendung von Sitzungsinformationen, um den Authentifizierungsstatus eines Nutzers während seiner Interaktion mit der Anwendung zu verfolgen. Sobald sich ein Nutzer erfolgreich authentifiziert hat, wird eine Sitzung erstellt und es wird dem Nutzer ein eindeutiger Session-Identifier zugewiesen. Dieser Session-Identifier dient als temporärer Identifikator und wird bei jeder weiteren Anfrage an die Anwendung mitgeschickt. Während der Sitzung behält die Anwendung den Authentifizierungsstatus des Nutzers bei und kann anhand des Session-Identifiers den Nutzer identifizieren, ohne dass er sich bei jeder Anfrage erneut authentifizieren muss. Das Session-Handling stellt somit sicher, dass nur authentifizierte Nutzer Zugriff auf geschützte Ressourcen und Funktionen der Anwendung erhalten, solange ihre Sitzung aktiv ist [80]. O.Sess_1 (vgl. Anhang A.6) soll sicherstellen, dass das Session-Handling durch ein sicheres Framework bereitgestellt wird. Dabei verweist das BSI auf den Prüfaspekt O.Ntwk_3 (siehe Abschnitt 4.9.1).

Folgende Anforderungen muss der Session-Identifier erfüllen:

- Der Session-Identifier soll mit einem sicheren Zufallsgenerator des Hintergrundsystems erzeugt werden (O.Sess_2, s. A.6). Die Verwendung und Empfehlungen von sicheren Zufallsgeneratoren wurde bereits in Abschnitt 4.5.4 vorgenommen.
- Der Session-Identifier muss als sensibles Datum eingestuft werden (O.Sess_3, s. A.6). Er unterliegt somit den Anforderungen an sensiblen Daten (siehe unter anderem Abschnitt 4.1.3 und 4.2.2).
- Der Session-Identifier darf nicht unverschlüsselt auf permanenten Speichermedien abgelegt werden (O.Sess_4, s. A.6). Diese Anforderung kann mittels den in Abschnitt 4.5.1 vorgestellten Verschlüsselungsverfahren umgesetzt werden.

- Es muss dem Nutzer ermöglicht werden ein oder alle zuvor ausgestellten Session-Identifier ungültig machen zu können (O.Sess_5, s. A.6). Dies kann, neben Log-out Funktion und automatisches Beenden der Anwendung, durch eine Session-Verwaltung in den Einstellungen umgesetzt werden. Dort wird dem Nutzer eine Übersicht über seine aktiven Sitzungen oder Session-Identifier angezeigt. Von dort aus kann der Nutzer einzelne Sitzungen manuell beenden, indem er sie als ungültig markiert.

- Der Session-Identifier muss nach dem Beenden der Anwendung sicher gelöscht und das Hintergrundsystem über das Beenden der Session informiert werden (O.Sess_-6, s. A.6). Die Umsetzung und Empfehlungen in Bezug auf das sichere Löschen können dem Abschnitt 4.1.2 entnommen werden. Zu beachten ist, dass der Session-Identifier nicht nur bei der aktiven Beendigung der Anwendung durch den Nutzer (Log-out), sondern auch durch das automatische Beenden der Anwendung sicher gelöscht werden muss. Mit automatischen Beenden sind das Ablaufen der zeitlichen Fristen in Bezug auf Hintergrundbetrieb, aktiver Nutzung (active time) und inaktiver Nutzung (idle time), welche im Abschnitt 4.6.3 thematisiert worden sind, gemeint.

Fazit Diese Prüfaspekte beziehen sich auf das Session-Handling und den sicheren Umgang mit Session-Identifier. In dem Prüfaspekt O.Sess_6 referenziert das BSI in Bezug auf das automatische Beenden nur O.Auth_8 und O.Auth_9, also das Beenden der Anwendung nach einer gewissen Zeit der aktiven (active time) und der inaktiven (idle time) Nutzung. O.Auth_7 sollte aber auch berücksichtigt werden, denn auch wenn die Anwendung eine gewisse Zeit im Hintergrundbetrieb lief, wird diese automatisch beendet. Dementsprechend sollte dieser Prüfaspekt für die Erfüllung von O.Sess_6 auch berücksichtigt werden.
Generell wird Beenden der Anwendung in diesen Prüfaspekten gleichgesetzt mit der Bedeutung des Abmeldens (Log-out). Sobald also der Nutzer nicht mehr angemeldet ist und sich neu Authentisieren muss, gilt die Anwendung als beendet.

4.6.10 Prüfaspekte O.Tokn_1, O.Tokn_2, O.Tokn_3, O.Tokn_4, O.Tokn_5 und O.Tokn_6

Diese Prüfaspekte definieren die Anforderungen an Authentifizierungstoken als zustandslose Authentifizierungsmaßnahme.

Ein Authentifizierungstoken kann als ein kompaktes Datenpaket gesehen werden, das alle erforderlichen Informationen zur Authentifizierung eines Nutzers oder Geräts enthält. Es ermöglicht den Zugriff auf geschützte Ressourcen, ohne dass ein spezifischer Zustand (beispielsweise Sitzungsinformationen) auf dem Server gespeichert werden muss. Das Authentifizierungstoken wird typischerweise direkt von der Anwendung erstellt und bei jeder Anfrage mitgeschickt [89].

O.Tokn_1 (vgl. Anhang A.6) legt fest, dass das Authentifizierungstoken in einem sicheren Speicherbereich wie dem KeyStore (für Android) [29] oder dem KeyChain (für iOS) [51] auf dem Endgerät gespeichert werden soll. Diese Speicherbereiche bieten Schutz vor unbefugtem Zugriff und sind speziell für die sichere Verwaltung von kryptographischen Schlüsseln und sensiblen Daten entwickelt worden.

Das Authentifizierungstoken soll so strukturiert werden, dass es nur die spezifischen Felder enthält, die von der Anwendung erwartet werden (O.Tokn_3, s. A.6). Nicht relevante Informationen sollen aus dem Authentifizierungstoken entfernt werden, um unnötige Sicherheitsrisiken zu vermeiden. Dies betrifft vor allem sensible Daten, diese dürfen unter keinen Umständen in ein Authentifizierungstoken eingebettet sein (O.Tokn_2, s. A.6).

Um die Anforderung O.Tokn_4 (vgl. Anhang A.6) zu erfüllen, darf der private Schlüssel, der zum Signieren des Authentifizierungstoken verwendet wird, nicht auf dem Gerät vorliegen. Das Gerät darf nicht in der Lage sein, eigene Authentifizierungstoken zu signieren.

Es muss dem Nutzer ermöglicht werden ein oder alle zuvor ausgestellten Authentifizierungstoken ungültig zu machen (O.Tokn_5, s. A.6). Dies könnte durch eine zentrale Token-Verwaltung auf dem Server oder über eine spezielle API ermöglicht werden. Dadurch kann der Nutzer beispielsweise bei einem verlorenen oder gestohlenen Gerät alle bestehenden Token deaktivieren und somit den Zugriff von unbefugten Personen verhindern.

Gleichnamig O.Sess_6 (siehe 4.6.9) muss nach dem Beenden der Anwendung das Authentifizierungstoken sicher gelöscht und das Hintergrundsystem über das Beenden informiert

werden (O.Tokn_6, s. A.6). Die Umsetzung und Empfehlungen können für Authentifizierungstoken analog von O.Sess_6 (Abschnitte 4.6.9 und 4.6.9) befolgt werden.

Fazit Der Aufbau der Anforderungen an Authentifizierungstoken sind ähnlich definiert und strukturiert wie die an die Session-Identifier. Bei den Authentifizierungstoken muss besonders darauf geachtet werden, welche Felder benötigt werden und dass keine sensiblen Daten in den Token eingebettet werden.

4.6.11 Bewertung Authentifizierung

Bewertungsaspekt → Prüfaspekt ↓	Aufwand	Umsetz-barkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
O.Auth_1, O.Auth_2, O.Auth_3, O.Auth_4	hoch	schwer	mittel	hoch
O.Auth_5, O.Auth_6	mittel	leicht	eindeutig	mittel
O.Auth_7, O.Auth_8, O.Auth_9, O.Auth_10, O.Auth_12	leicht	mittel	ungenau	hoch
O.Auth_11	mittel	schwer	ungenau	mittel
O.Pass_1, O.Pass_2, O.Pass_3, O.Pass_4, O.Pass_5	mittel	leicht	mittel	mittel
O.Biom_1, O.Biom_2, O.Biom_3, O.Biom_4	hoch	mittel	ungenau	mittel
O.Biom_5, O.Biom_6	mittel	mittel	mittel	mittel
O.Biom_7, O.Biom_8	mittel	mittel	ungenau	mittel
O.Sess_1, O.Sess_2, O.Sess_3, O.Sess_4, O.Sess_5, O.Sess_6	hoch	mittel	mittel	mittel
O.Tokn_1, O.Tokn_2, O.Tokn_3, O.Tokn_4, O.Tokn_5, O.Tokn_6	hoch	mittel	mittel	mittel
Gesamtbewertung	🟡	🟡	🟡	🔴

Tabelle 4.6: Bewertung Authentifizierung

4.7 Datensicherheit

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Datensicherheit fallen.
Die darin enthaltenen Prüfaspekte legen Anforderungen für die sichere Handhabung von Daten in DiGA fest. Hierbei werden Aspekte wie Verschlüsselung, Schutz vor unautorisiertem Zugriff, Datenverarbeitung im Hintergrund und sichere Deinstallation betont,

um die Vertraulichkeit, Integrität und Verfügbarkeit sensibler personenbezogener Informationen sicherzustellen.

4.7.1 Prüfaspekte O.Data_1, O.Data_2, O.Data_3, O.Data_4, O.Data_7, O.Data_12, O.Data_13, O.Data_14 und O.Data_15

Diese Prüfaspekte adressieren Anforderungen an die Verschlüsselung von Daten, die sichere Speicherung und Verarbeitung sensibler Informationen sowie den Schutz vor Datenzugriff durch Dritte oder in gesperrtem Zustand des Endgeräts.

O.Data_1 (vgl. Anhang A.7) formuliert allgemein die Anforderung, dass die Werkseinstellungen der Anwendung maximale Sicherheit bieten müssen. Das bedeutet, dass die Anwendung bereits von Anfang an so konfiguriert sein sollte, dass bei der Installation und erstmaligen Nutzung der Anwendung die voreingestellten Sicherheitsmaßnahmen und Datenschutzrichtlinien bereits aktiv sind, ohne dass der Nutzer zusätzliche Einstellungen vornehmen muss. Der Evaluator wird diese Standardeinstellungen der Anwendung bei Installation überprüfen. Fokus liegt hier auch auf den von der Anwendung geforderten Berechtigungen des Betriebssystems. Diese dürfen nur angefragt werden, wenn sie dem Zweck der Anwendung entsprechen (siehe Abschnitt 4.1) und erst an dem Punkt angefragt werden, an dem sie verwendet werden.

Neben dieser allgemeinen Anforderung beziehen sich die weiteren Prüfaspekte primär auf sensible Daten und werden im Folgenden zusammengefasst:
O.Data_2 und O.Data_3 (vgl. Anhang A.7) legen fest, dass sensible Daten nicht unverschlüsselt gespeichert werden dürfen. Weiterführend soll die Speicherung von sensiblen Daten in geschützten Umgebungen stattfinden, welche vor Einsicht und Manipulation geschützt sind. Es ist wichtig zu beachten, dass auch das Schlüsselmaterial sicher in einer hardwareunterstützten Umgebung verwaltet werden soll. Das BSI schlägt hierfür beispielsweise Embedded Secure Element oder Trusted Execution Environment vor, welche bereits in Abschnitt 4.5.3 vorgestellt worden sind. Generell soll sowohl die Speicherung, als auch die Verarbeitung sensibler Daten nur im Hintergrundsystem erfolgen (O.Data _7, s. A.7). Zusätzlich muss die Anwendung sicherstellen, dass im gesperrten Zustand des Endgeräts alle sensiblen Daten verschlüsselt sind (O.Data_14, s. A.7). Zu beachten ist außerdem, dass in Bezug auf die Speicherung nicht zwischen flüchtigen (zum Beispiel Arbeitsspeicher) und dauerhaften Speichern unterschieden wird. Dritte dürfen

gemäß O.Data_4 nicht auf Ressourcen zugreifen, die einen Zugriff auf sensible Daten ermöglichen. Dieser Prüfaspekt ist ähnlich dem bereits in Abschnitt 4.4.1 behandelten Prüfaspekt O.TrdP_6.
Lokal gespeicherte Daten - unabhängig davon, ob es sich um sensible Daten handeln - müssen sicher an das Gerät gebunden werden (O.Data_15, s. A.7). Beispielsweise wird eine hardwarebasierte Gerätebindung durch die Nutzung von Android KeyStore [29] und iOS KeyChain [51] durchgeführt. Es besteht die Ausnahme, dass der Nutzer explizit Daten auf ein anderes Gerät exportieren möchte, um dieses für die Anwendung nutzen zu können. In diesem Fall ist der Export der Daten erlaubt, unter der Bedingung, dass der Nutzer sich ausreichend authentifiziert. Der genannte Export von Daten wird von dem Prüfaspekt O.Data_12 eingeschränkt. Es dürfen keine sensiblen Daten wie biometrische Daten oder private Schlüssel aus der Quelle beziehungsweise Komponente exportiert werden, in der sie erzeugt wurden. Bei diesen Daten handelt es sich nämlich um sensible Daten, für die keine Notwendigkeit für einen Export besteht.
Wenn die Anwendung im Betrieb sensible Daten anzeigt, muss ein Speichern oder Zugriff durch Dritte verhindert werden (O.Data_13, s. A.7). Dementsprechend muss die Anwendung das Erzeugen von Screenshots verbieten. Das umfasst nicht nur die aktive Erzeugung eines Screenshots, sondern bezieht sich auch auf das passive Erzeugen durch den Task-Manager. Auch eine Vorschau der Anwendung im App Switcher sollte geschwärzt werden, um so dem Risiko zu entgehen, ungewollt sensible Daten Dritten zugänglich zu machen.

Fazit In diesem gesamten Prüfaspekt sind teilweise die Formulierungen des BSI nicht gut gewählt. Beispielsweise O.Data_3 verwendet den Begriff 'ablegen' anstatt direkt von 'speichern' zu sprechen. Ein anderes Beispiel ist O.Data_5 'dürfen nicht in der Anwendung gehalten werden', hier wäre es ebenfalls zu empfehlen, von Löschung zu sprechen. Solche Formulierungen führen dazu, dass Missverständnisse entstehen können, eine klare präzise Formulierung wäre besser.
Außerdem beschreiben diese Anforderungen sehr ähnliche Themen und es wirkt als seien einige redundant, insbesondere in Bezug auf bereits formulierte Prüfaspekte in anderen Kapiteln. Es wäre also besser gewesen, einen ausführlicheren Prüfaspekt zu erstellen mit den entsprechenden Verweisen auf die jeweiligen anderen Prüfaspekte.
Generell gilt, die Verarbeitung und Speicherung der sensiblen Daten muss verschlüsselt und in sicheren Bereichen stattfinden. Ressourcen sollten vor dem Zugriff Dritter geschützt werden und verwendetes Schlüsselmaterial muss ebenfalls verschlüsselt und in

sicherer Umgebung aufbewahrt werden. Ein Export von Daten und vor allem sensiblen Daten darf nur erfolgen, wenn die Notwendigkeit des Exportes besteht und der Nutzer sich entsprechend authentifiziert. Außerdem müssen Funktionen wie Screenshots und Darstellung der Anwendung im App Switcher bedacht und in Bezug auf sensible Daten verhindert werden. Eine Strategie für das Einschränken von Screenshots sollte generell für alle Daten umgesetzt werden, insofern es nicht für den Anwendungszweck erforderlich ist.

4.7.2 Prüfaspekte O.Data_5 und O.Data_6

Diese Prüfaspekte weisen erneut auf die Anforderungen an die Einhaltung des Anwendungszwecks hin. Die Grundsätze der Datensparsamkeit und der Zweckbindung können dem Abschnitt 4.1 entnommen werden, ebenso die Umsetzungsvorschläge.

Die Anwendung sollte nur die minimal erforderlichen Daten erheben und speichern, die für ihre Funktionen und den vorgesehenen Zweck notwendig sind (O.Data_6, s. A.7). Es ist wichtig, die Menge an gesammelten Daten zu begrenzen und sicherzustellen, dass diese nur für den spezifischen Zweck verwendet werden, für den sie erhoben wurden. Die Anwendung sollte keine sensiblen Daten sammeln, die nicht direkt für ihre Funktionen erforderlich sind. Ergänzend dazu sollte die Anwendung eine Datenverwaltung implementieren, die sicherstellt, dass sensible Daten nur so lange in der Anwendung gespeichert werden, wie sie tatsächlich benötigt werden. Nachdem die Daten für ihren vorgesehenen Zweck verwendet wurden, sollten sie unverzüglich gelöscht werden (O.Data_5, s. A.7).

Fazit Diese Prüfaspekte beziehen sich erneut darauf, dass unbedingt darauf geachtet werden muss, dass nur Daten erhoben werden, die für die Erfüllung des Zwecks notwendig sind und auch nur so lange gespeichert werden, wie sie benötigt werden. Es soll ausdrücken, wie wichtig es ist, dass möglichst wenig Daten erhoben werden. Empfehlungen wie die Dokumentation erfolgen kann, ist bereits in 4.1 erfolgt und wird aus diesem Grund hier nicht weiter thematisiert.

4.7.3 Prüfaspekte O.Data_8 und O.Data_9

Beide Prüfaspekte legen den Umgang mit der Verwendung von Aufnahmegeräten wie Kameras fest:

- O.Data_8 (vgl. Anhang A.7) besagt, dass bei der Verwendung von Aufnahmegeräten wie Kameras alle Metadaten mit Datenschutzrelevanz entfernt werden müssen. Metadaten sind zusätzliche Informationen, die in Dateien gespeichert werden und Rückschlüsse auf den Aufnahmeort, die eingesetzte Hardware und andere relevante Informationen zulassen könnten. Um die Preisgabe von nicht notwendigen personenbezogenen Informationen zu verhindern, muss die Anwendung in der Lage sein, die Metadaten zu analysieren und nicht notwendige Informationen zu entfernen.
- O.Data_9 (vgl. Anhang A.7) fordert, dass bei der Erhebung von sensiblen Daten durch Aufnahmegeräte wie Kameras verhindert werden muss, dass andere Anwendungen darauf zugreifen können. Es muss also verhindert werden, dass diese sensiblen Daten in öffentliche Verzeichnisse gespeichert werden. Beispielsweise darf keine Speicherung der sensiblen Daten in der Mediengalerie stattfinden, da auch andere Anwendungen auf diese Zugriff haben.

Fazit Es ist wichtig, darauf zu achten, dass die Metadaten und der Speicherort bei Aufnahmefunktionen des Gerätes geprüft werden. Dass Metadaten und aufgenommene sensible Daten nicht in öffentlichen Verzeichnissen (wie Mediengalerie) gespeichert werden, kann schnell bei der Umsetzung nicht bedacht werden, beziehungsweise wird es nicht so einfach als kritisch identifiziert.

4.7.4 Prüfaspekte O.Data_10 und O.Data_11

Diese Prüfaspekte sollen die Eingabe sensibler Daten durch den Nutzer sichern. Es ist durchaus ein Use Case, dass der Nutzer von der Anwendung aufgefordert wird, sensible Daten einzugeben, um den Anwendungszweck erfüllen zu können. Diese Benutzereingabe kann zum Beispiel über die Tastatur erfolgen. Um diesen Prozess zu sichern, stellt das BSI zwei Anforderungen:

1. Die Eingabe der sensiblen Daten soll für Dritte nicht erkennbar sein (O.Data_10, s. A.7)
2. Bei Eingabe sensibler Daten soll der Export in die Zwischenablage nicht möglich sein (O.Data_11, s. A.7)

In Bezug auf den ersten Punkt, dass die Eingabe für Dritte nicht erkennbar sein soll, prüft das BSI, ob Softwaretastaturen des Betriebssystems oder von Drittanbietern zur Eingabe sensibler Daten verwendet werden können. Damit verbundene Caches sowie Autokorrektur- und Autovervollständigungsverfahren können von Dritten ausgewertet werden, sodass ungewollt sensible Daten an Dritte gelangen. Es wird leider an dieser Stelle nicht deutlich, welche Maßnahmen erwartet werden, es klingt fast so, als sollen Softwaretastaturen des Betriebssystems nicht erlaubt werden, was den Implementierungsaufwand deutlich erhöhen würde. Das hätte zur Folge, dass nur selbst implementierte Softwaretastaturen verwendet werde dürfen. Alternativ könnte man die Formulierung so interpretieren, dass Softwaretastaturen von Drittanbietern nicht zugelassen werden sollen, sondern nur Softwaretastaturen des Betriebssystems. Sollte dieser Weg gewählt werden, sollten bei der Nutzung von Softwaretastaturen des Betriebssystem noch weitere Mechanismen implementiert werden, um die Sicherheit der Eingabe sensibler Daten zu gewährleisten. Dazu gehören Funktionen wie Eingabe-Maskierung (keine Anzeige im Klartext), Cache-Vermeidung und die Deaktivierung von Autokorrektur- und Autovervollständigungsverfahren.

Die zweite Anforderung soll verhindern, dass sensible Daten versehentlich von anderen Anwendungen abgerufen werden können. Eine Möglichkeit zur Umsetzung besteht darin, die Clipboard-Funktion in der Anwendung zu deaktivieren oder eine eigene geschützte Zwischenablage zu implementieren, auf die nur die Anwendung selbst zugreifen kann. Dadurch wird verhindert, dass die eingegebenen sensiblen Daten ungeschützt in der Zwischenablage liegen und von anderen Anwendungen abgerufen werden können.

Fazit Beide Prüfaspekte betreffen den Umgang mit Daten während der Benutzereingabe und fordern, dass die Anwendung Maßnahmen ergreift, um das Auslesen oder die Aufzeichnung der sensiblen Daten zu verhindern und so die Vertraulichkeit der Informationen zu wahren. Besonders bei dem Prüfaspekt O.Data_10 wird deutlich, dass das BSI mehr Informationen über ihre Erwartungshaltung liefern sollte, damit die Anforderungen der Prüfaspekte umgesetzt und erfüllt werden können.

4.7.5 Prüfaspekte O.Data_16 und O.Data_19

Diese Prüfaspekte fokussieren sich auf die Anforderungen an die Anwendung bei Diebstahl oder sonstigem Geräteverlust.

O.Data_16 (vgl. Anhang A.7) bezieht sich auf die Verwendung von nicht vor Diebstahl geschützte Speichermedien wie beispielsweise unverschlüsselte SD-Karten. Die Anwendung soll dann beim Zugriff auf das Speichermedium eine Warnung anzeigen, die den Nutzer über das potenzielle Risiko informiert. Dies kann beispielsweise durch eine Popup-Nachricht oder eine Benachrichtigung geschehen.

O.Data_19 (vgl. Anhang A.7) bietet der Anwendung die Möglichkeit eine Funktion zu implementieren, die es dem Nutzer ermöglicht, den Kill-Switch [90] auszulösen, wenn das Gerät als verloren oder gestohlen gemeldet wird. Durch das Auslösen des Kill-Switches werden alle sensiblen Daten auf Anwendungsebene sicher gelöscht oder unzugänglich gemacht, um eine unbefugte Nutzung zu verhindern. Bevor der Kill-Switch aktiviert wird, muss das Hintergrundsystem eine starke Authentifizierung des Nutzers durchführen, um eine missbräuchliche Nutzung zu verhindern.

Fazit Das BSI fordert bei der Nutzung von nicht Diebstahl geschützten Speichermedien nur einen Hinweis für den Nutzer. Um auf der sicheren Seite zu sein, könnte man diese Anforderung noch um die Empfehlung erweitern, einen Consent des Nutzers einzufordern und diesen zu protokollieren. Die Umsetzung des Kill-Switches ist lediglich eine optionale Empfehlung und muss nicht zwingend umgesetzt werden, da es aber die Sicherheit der Nutzerdaten erhöht, sollte die Umsetzung in Betracht gezogen werden.

4.7.6 Prüfaspekte O.Data_17 und O.Data_18

Die Prüfaspekte O.Data_17 und O.Data_18 (vgl. Anhang A.7) definieren wie mit sensiblen Daten und anwendungsspezifischen Anmeldeinformationen bei der Deinstallation umgegangen werden soll. Und zwar sollen diese Daten und Informationen nach der Deinstallation auf dem Endgerät nicht mehr zugreifbar sein und die der Nutzer soll die Möglichkeit haben, die Daten vollständig löschen zu lassen.

Fazit Die Prüfaspekte sind so formuliert, dass bei der Deinstallation die sensiblen Daten und anwendungsspezifischen Anmeldeinformationen nur auf Wunsch des Nutzers gelöscht werden sollen und ansonsten nur nicht mehr zugreifbar gemacht werden. Generell sollten, aber im Falle einer Deinstallation ohnehin alle sensiblen Daten und anwendungsspezifischen Anmeldeinformationen vollständig gelöscht werden.

4.7.7 Bewertung Datensicherheit

Bewertungsaspekt → Prüfaspekt ↓	**Aufwand**	**Umsetz-barkeit**	**Präzision der BSI Anforderung**	**Höhe der Priorität im Gesamtprojekt**
O.Data_1, O.Data_2, O.Data_3, O.Data_4, O.Data_7, O.Data_12, O.Data_13, O.Data_14, O.Data_15	hoch	mittel	ungenau	hoch
O.Data_5, O.Data_6	hoch	mittel	mittel	mittel
O.Data_8, O.Data_9	hoch	mittel	mittel	hoch
O.Data_10, O.Data_11	hoch	schwer	ungenau	mittel
O.Data_16, O.Data_19	mittel	mittel	mittel	mittel
O.Data_17, O.Data_18	leicht	leicht	ungenau	niedrig
Gesamtbewertung				

Tabelle 4.7: Bewertung Datensicherheit

4.8 Kostenpflichtige Ressourcen

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Kostenpflichtige Ressourcen fallen. Die darin enthaltenen Prüfaspekte legen Anforderungen für die sichere und transparente Gestaltung von Kosten und Bezahlvorgängen in DiGA fest. Hierbei werden Aspekte wie Nutzereinverständnis, Widerrufsmöglichkeiten von Einverständnissen, Verwaltung von Transaktionshistorien und die Sicherheit von Zahlungsströmen betont, um eine vertrauenswürdige und transparente Nutzung kostenpflichtiger Ressourcen und Leistungen zu gewährleisten.

4.8.1 Prüfaspekte O.Paid_1, O.Paid_2, O.Paid_3, O.Paid_4 und O.Paid_5

Diese Prüfaspekte legen Wert auf Transparenz und Einverständniserklärung des Nutzers in Bezug auf kostenpflichtige Leistungen und Ressourcen.

Kostenpflichtige Leistungen (Zusatzfunktionen, Premiumzugriffe etc.) sowie kostenpflichtige Ressourcen (SMS, Telefonate, mobile Daten etc.) müssen für den Nutzer deutlich kenntlich gemacht werden (O.Paid_1, s. A.8). Sinnvoll ist es bereits vor der Installation der Anwendung, also im jeweiligen App Store, aus dem die Anwendung bezogen werden kann, einen ersten Hinweis einzubauen, dass die Anwendung kostenpflichtige Inhalte enthält und diese direkt auflisten. Im Betrieb der Anwendung muss vor der Nutzung der kostenpflichtigen Ressource informiert werden, dass es sich um einen kostenpflichtigen Inhalt handelt. Diese Information soll vor der Verwendung der kostenpflichtigen Leistungen auch mit einer Einholung des Einverständnisses des Nutzers gekoppelt werden. Folgende Anforderungen gelten für die Einverständniserklärung des Nutzers:

- Kostenpflichtige Leistungen dürfen erst nach dem aktiven Einverständnis (Bestätigung) des Nutzers erbracht werden (O.Paid_2, s. A.8).
- Das Einverständnis ist vor einer Zugriffsanforderung (beispielsweise Android-Berechtigungen) auf kostenpflichtigen Ressourcen einzuholen. Die Nutzung von Diensten, die zusätzliche Kosten verursachen, dürfen erst nach der aktiven Abgabe des Einverständnisses möglich sein (O.Paid_3, s. A.8).
- Der Nutzer muss jederzeit in der Lage sein, zuvor erteilte Einverständnisse zurückzuziehen(O.Paid_5, s. A.8). Hierfür ist dem Nutzer eine Liste mit seinen getätigten Einverständnissen anzuzeigen, in der er die Möglichkeit hat, diese nachträglich zu ändern.
- Für den Fall, dass kostenpflichtige Inhalte häufig oder regelmäßig verwendet werden müssen, kann auch ein dauerhaftes Einverständnis vom Nutzer eingeholt werden. In diesem Fall wird geprüft, ob die Häufigkeit der Nutzung der kostenpflichtigen Leistung dem primären Zweck der Anwendung (siehe 4.1.1) entspricht (O.Paid_4, s. A.8).

Wie ein Aufbau eines Einwilligungsverzeichnisses aussehen könnte, kann dem Abschnitt 4.1.2 entnommen werden. Die dortigen Inhalte beziehen sich auf die Einverständniserklärung in Bezug auf die Datenerhebung, können aber an diesen Use Case angepasst werden.

Fazit Transparenz hinsichtlich kostenpflichtiger Ressourcen ist also eine Grundvoraussetzung um die Anforderungen zu erfüllen. Die Anwendung muss dem Nutzer klare Informationen bereitstellen und ihm die Möglichkeit geben, seine Zustimmung zu geben

oder zurückzuziehen. Bei diesen Prüfaspekten ist hinsichtlich Einwilligung ähnlich zu verfahren wie bei den Einwilligungen des Prüfaspektes Anwendungszweck.

4.8.2 Prüfaspekte O.Paid_6 und O.Paid_8

Sollte ein Nutzer kostenpflichtige Inhalte in Anspruch nehmen, müssen die getätigten Transaktionen und Kosten dokumentiert werden. Für die Transaktionen soll eine sogenannte Transaktionshistorie erstellt werden (O.Paid_6, s. A.8). Eine Transaktionshistorie ist eine chronologische Aufzeichnung von Transaktionen, die in einer Anwendung durchgeführt werden. Die Transaktionshistorie enthält Informationen über die einzelnen Zahlungsvorgänge, die der Nutzer getätigt hat. Die Historie zeigt typischerweise das Datum, die Uhrzeit, den Betrag und die Art der Transaktion an. Sie dient dazu, dem Nutzer eine Übersicht über seine getätigten Zahlungen und Ausgaben zu geben und eventuell auch als Nachweis für getätigte Käufe zu fungieren. Laut BSI soll die Transaktionshistorie als ein sensibles Datum eingestuft werden, soll daher sicher im Hintergrundsystem und nicht in der Anwendung gespeichert werden und muss den Anforderungen des Prüfaspektes O.Purp_8 (siehe Abschnitt 4.1.3) genügen.
Der Nutzer soll die Möglichkeit haben, eine Übersicht der entstandenen Kosten einzusehen (O.Paid_8, s. A.8). In dieser Übersicht sind alles entstandenen Kosten aufzulisten, auch für einzelne Zugriffe. Die Informationen dieser Übersicht basiert auf den Inhalten der Transaktionshistorie.

Fazit Bei der Transaktionshistorie ist unbedingt darauf zu achten, dass es in allen Fällen wie ein sensibles Datum behandelt wird. Das bezieht sich nicht nur auf den vom BSI benannten Prüfaspekt O.Purp_8, sondern auch auf die weiteren, die Anforderungen an sensible Daten stellen. Demnach sind nicht nur Anwendungszweck und das Teilen mit Dritten in Bezug auf die Transaktionshistorie zu beachten, sondern auch beispielsweise das sichere Speichern und die Verschlüsselung. Die Informationen der Zahlungsübersicht für den Nutzer basieren auf den Informationen der Transaktionshistorie. Auch wenn zur Erstellung der Übersicht nicht die alle Daten aus der Transaktionshistorie entnommen werden, sind Teile der Informationen in der Zahlungsübersicht enthalten. Das BSI definiert in der TR-03161 [58] nicht, dass sich die Zahlungsübersicht auch um ein sensibles Datum handelt. Da sich die Daten aber ähneln und Rückschlüsse auf das Transaktionsverhalten des Nutzers geben können, sollte für die Zahlungsübersicht auch abgewägt werden, ob es nicht auch als sensibles Datum behandelt werden sollte.

4.8.3 Prüfaspekte O.Paid_7, O.Paid_9 und O.Paid_10

Diese Prüfaspekte beziehen sich auf Sicherheit des Zahlungsprozesses. Um die Zahlung durchzuführen, erlaubt der Prüfaspekt O.Paid_10 (vgl. Anhang A.8) die Verwendung von Zahlverfahren von Drittanbietern unter der Voraussetzung, dass die Anforderungen an Drittanbieter-Software (siehe 4.4 von dem Drittanbieter erfüllt sind. Dies bedeutet, dass Zahlungsplattformen oder -dienstleister, die von der Anwendung genutzt werden, hohen Sicherheitsstandards entsprechen müssen, um eine sichere Zahlungsabwicklung zu gewährleisten.
Weiterführend muss die Zahlungsabwicklung und die damit verbundenen Transaktionen in der Anwendung geschützt und verschlüsselt sein. Das legt der Prüfaspekt O.Paid_7 (vgl. Anhang A.8) fest, um zu verhindern, dass Dritte in der Lage sind, Zahlungsströme zurückverfolgen können und anhand dessen Rückschlüsse auf die Eigenschaften oder das Verhalten des Nutzers möglich sind (Profilbildung). Zur Erfüllung dieses Prüfaspektes soll ein Konzept vorgelegt werden, das sicherstellt, dass Dritte die Zahlungsströme zur Nutzung von Anwendungsfunktionen nicht zurückverfolgen können. Um die Abwicklung der Zahlung noch weiter zu sichern, legt der Prüfaspekt O.Paid_9 (vgl. Anhang A.8) fest, dass die Validierung von getätigten Bezahlvorgängen im Hintergrundsystem stattfinden sollte, um Manipulationen oder Fälschungen zu verhindern.

Fazit Neben dem Einverständnis zur Nutzung von kostenpflichtigen Leistungen und das Führen einer Übersicht und Transaktionshistorie, fokussieren sich diese Prüfaspekte auf den eigentlichen Ablauf des Zahlungsprozesses. Die Erfüllung dieser Anforderungen bietet Schutz vor unbefugtem Zugriff auf Zahlungsströme, die sichere Validierung von Bezahlvorgängen und die Einhaltung hoher Sicherheitsstandards für Zahlungsdienstleister und Plattformen.

4.8.4 Bewertung Kostenpflichtige Ressourcen

Bewertungsaspekt → Prüfaspekt ↓	Aufwand	Umsetzbarkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
O.Paid_1, O.Paid_2, O.Paid_3, O.Paid_4, O.Paid_5	mittel	leicht	eindeutig	niedrig
O.Paid_6, O.Paid_8	mittel	leicht	ungenau	niedrig
O.Paid_7, O.Paid_9, O.Paid_10	mittel	mittel	mittel	mittel
Gesamtbewertung	🟡	🟢	🟡	🟢

Tabelle 4.8: Bewertung Kostenpflichtige Ressourcen

4.9 Netzwerkkommunikation

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Netzwerkkommunikation fallen. Die darin enthaltenen Prüfaspekte legen Anforderungen für die sichere und verlässliche Kommunikation von DiGA im Netzwerk fest. Hierbei werden Aspekte wie Verschlüsselung, Validierung von Zertifikaten, Integritätssicherung von Daten und die Nutzung sicherer Kommunikationskanäle betont, um eine geschützte und zuverlässige Datenübertragung zu gewährleisten.

4.9.1 Prüfaspekte O.Ntwk_1, O.Ntwk_2, O.Ntwk_3 und O.Ntwk_7

Diese Prüfaspekte zielen es darauf ab die Sicherheit der Netzwerkkommunikation sicherzustellen durch die Verwendung von TLS-Verschlüsselung, die Einhaltung der aktuellen Best Practices für TLS-Konfiguration und die Verwendung sicherer Kommunikationskanäle durch sichere Frameworks oder Betriebssystemfunktionen. Hierfür sollen folgende Maßnahmen ergriffen werden:

- Jede Kommunikation zwischen der Anwendung und anderen Komponenten muss durchgängig mit TLS verschlüsselt sein (O.Ntwk_1, s. A.9).

- Die TLS-Konfiguration der zuvor benannten TLS-Verbindungen von O.Ntwk_1 soll dem aktuellem Stand der Technik entsprechen (O.Ntwk_2, s. A.9). Hierfür verweist das BSI auf die Technische Richtlinie TR-02102-2 [63]. In dieser Richtlinie werden sichere TLS-Konfigurationen vorgestellt für sowohl TLS 1.2, als auch TLS 1.3. In dem Abschnitt 4.5.2 wurde diese Technische Richtlinie und der darin beschriebene Stand der Technik bereits beschrieben.
- O.Ntwk_3 (vgl. Anhang A.9) beschreibt die Möglichkeiten zum Aufbauen sicherer Kommunikationskanäle. Ein sicherer Kommunikationskanal bezeichnet eine verschlüsselte und geschützte Verbindung zwischen der Anwendung und einer anderen Komponente, um sicherzustellen, dass Daten während der Übertragung vor unbefugtem Zugriff geschützt sind. Sichere Kommunikationskanäle sollten vorzugsweise mit der Sicherheitsfunktionaliät der jeweiligen Betriebssystem Plattform aufgebaut werden, alternativ kann aber auch sicherheitsüberprüfte Drittsoftware verwendet werden. Eigene Implementierungen sind nicht zulässig. Zur Umsetzung unter Android kann die Anwendung die Network Security Configuration [30] verwenden, um TLS-Konfigurationen festzulegen. Für iOS kann die App Transport Security (ATS) [53] verwendet werden, um sichere Kommunikationskanäle aufzubauen.
- Um sicherzustellen, dass nur sicher TLS-verschlüsselte Kommunikationskanäle verwendet werden, müssen plattformspezifische Fallback-Mechanismen deaktiviert werden (O.Ntwk_7, s. A.9). Um die Fallback-Mechanismen zu deaktivieren, müssen entsprechende Konfigurationen in der Network Security Configuration für Android (Cleartext Traffic Opt-out deaktivieren) [30] und in der ATS [53] für iOS vorgenommen werden. Dies bedeutet, dass die Anwendung keine Ausnahmen für unverschlüsselte Kommunikation zulässt und immer auf sichere Kommunikationskanäle besteht.

Fazit Diese Prüfaspekte legen fest, dass jegliche Kommunikation TLS-verschlüsselt stattfinden muss und wie sichere Kommunikationskanäle aufzubauen sind. Einige dieser Inhalte sind redundant, da das BSI auf die TR-02102-2 [63] verweist, auf die bereits in dem Prüfaspekt O.Cryp_2 verwiesen worden ist. Es wäre durchaus zu empfehlen, dass das BSI bereits dort eine Verlinkung zu diesem Prüfaspekt O.Ntwk_2 herstellen würde, damit die vielen Querverweise unter den Prüfaspekten deutlich werden.

4.9.2 Prüfaspekte O.Ntwk_4, O.Ntwk_5 und O.Ntwk_6

Diese Prüfaspekte zielen es darauf ab Vertraulichkeit und Integrität der Kommunikation zu gewährleisten.
Der Prüfaspekt O.Nwk_4 (vgl. Anhang A.9) fordert, dass die Anwendung Zertifikats-Pinning unterstützen muss. Zertifikats-Pinning ist ein Sicherheitsmechanismus, bei dem die Anwendung das Server-Zertifikat des Hintergrundsystems überprüft und speichert. Die Anwendung legt vorab ein oder mehrere Server-Zertifikate fest, die als vertrauenswürdig gelten. Wenn die Anwendung eine Verbindung zum Hintergrundsystem (Server) herstellt, findet der SSL/TLS-Handshake statt. Während dieses Handshakes sendet der Server sein Zertifikat an die Anwendung. Die Anwendung überprüft das vom Server empfangene Zertifikat auf verschiedene Kriterien, um seine Gültigkeit und Vertrauenswürdigkeit zu bestätigen. Dazu gehören unter anderem die Gültigkeitsdauer, der Aussteller (Zertifizierungsstelle) und die Schlüssellänge des Zertifikats. Sie vergleicht das empfangene Zertifikat mit den zuvor festgelegten und gespeicherten Zertifikaten. Wenn das empfangene Zertifikat mit einem der gespeicherten Zertifikate übereinstimmt, wird die Verbindung als vertrauenswürdig akzeptiert und die Kommunikation kann stattfinden. Bei zukünftigen Verbindungen stellt die Anwendung sicher, dass nur das vorher festgelegte und gespeicherte Zertifikat akzeptiert wird. Andere von unbekannten oder unvertrauenswürdigen Zertifizierungsstellen ausgestellte Zertifikate, werden nicht akzeptiert [18].
Die Anforderung für O.Ntwk_5 (vgl. Anhang A.9), dass die Anwendung das Server-Zertifikat des Hintergrundsystems überprüfen muss ist mit der zuvor beschriebenen Umsetzung des Certificate Pinnings bereits erfüllt.
Neben der initialen Überprüfung der Vertrauenswürdigkeit des Hintergrundsystems, legt das BSI mit dem Prüfaspekt O.Ntwk_6 (vgl. Anhang A.9) weiterführend fest, dass die Anwendung auch die Integrität und Authentizität der Antworten des Hintergrundsystems validieren muss. Die Validierung der Antworten soll sicherstellen, dass die enthaltenen Daten nicht manipuliert wurden und tatsächlich vom Hintergrundsystem stammen. Um die erhaltenen Daten zu prüfen, kann die Anwendung digitale Signaturen oder Hashes verwenden, dessen Funktionsweise und weiterführende Empfehlungen in Abschnitt 4.5.1 beschrieben worden sind.

Fazit O.Ntwk_4 und O.Ntwk_5 sorgen dafür, dass Vertraulichkeit und Integrität der Kommunikation durch Zertifikats-Pinning und Überprüfung des Server-Zertifikats ge-

währleistet wird. O.Ntwk_6 befasst sich mit der Validierung der Integrität und Authentizität der Antworten des Hintergrundsystems, um sicherzustellen, dass die empfangenen Daten echt und unverändert sind.

4.9.3 Prüfaspekte O.Ntwk_8 und O.Ntwk_9

Diese beiden Prüfaspekte legen fest, dass Logging von Verbindungen und die Protokolllierung von Sicherheitsereignissen stattfinden soll und die resultierenden Dateien im Hintergrundsystem gespeichert werden sollen.
O.Ntwk_8 (vgl. Anhang A.9) stellt die Anforderung, dass sämtliche Netzwerkverbindungen, die von der Anwendung hergestellt werden, protokolliert werden müssen. Die Anwendung kann dafür eine umfangreiche Protokollierung dieser Netzwerkaktivitäten implementieren. Dabei werden beispielsweise Informationen wie Timestamp, Ziel-IP-Adresse, Port, Art der Verbindung und Fehlermeldungen erfasst. Gefordert ist mindestens die vollständige Erfassung des HTTP-Headers. Die Log-Dateien sollen auf dem Hintergrundsystem gespeichert werden und können bei Bedarf analysiert werden, um beispielsweise Sicherheitsvorfälle nachvollziehen zu können.
O.Ntwk_9 (vgl. Anhang A.9) erweitert das Logging durch die explizite Anforderung auch einen abgebrochenen Start der Anwendung als Sicherheitsereignis zu protokollieren und auf dem Hintergrundsystem zu speichern. Diese Anforderung ist wichtig zu betonen, denn wenn der Start der Anwendung aus bestimmten Gründen abgebrochen oder unterbrochen wird, kann dies auf mögliche Sicherheitsprobleme hinweisen, zum Beispiel auf einen Angriffsversuch oder auf unerwartetes Verhalten. Die Protokollierung sollte Informationen über alle ausgehenden Verbindungen enthalten sowie Metainformationen uber verwendete Proxys und uüberprüfte Server-Zertifikate. um eine ausführliche Post-Mortem Analyse zu ermöglichen.

Fazit Beide Prüfaspekte beziehen sich auf die Protokollierung von sicherheitsrelevanten Ereignissen im Hintergrundsystem. Sie dienen dazu, mögliche Sicherheitsvorfälle zu erkennen, zu dokumentieren und angemessen darauf zu reagieren. Die Protokollierung ermöglicht eine umfassende Nachverfolgung von Netzwerkverbindungen und Startvorgängen der Anwendung und trägt zur Sicherheit und Integrität bei.

4.9.4 Bewertung Netzwerkkommunikation

Bewertungsaspekt → Prüfaspekt ↓	Aufwand	Umsetz-barkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
O.Ntwk_1, O.Ntwk_3, O.Ntwk_7	hoch	mittel	ungenau	hoch
O.Ntwk_4, O.Ntwk_5, O.Ntwk_6	hoch	mittel	mittel	hoch
O.Ntwk_8, O.Ntwk_9	niedrig	leicht	eindeutig	mittel
Gesamtbewertung	🔴	🟡	🟡	🔴

Tabelle 4.9: Bewertung Netzwerkkommunikation

4.10 Plattformspezifische Interaktionen

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Plattformspezifische Interaktionen fallen. Die darin enthaltenen Prüfaspekte legen Leitlinien für die nahtlose Integration DiGA in die jeweiligen Betriebssystemplattformen fest.

4.10.1 Prüfaspekte O.Plat_1 und O.Plat_16

Die beiden Prüfaspekte befassen sich mit den Sicherheitseinstellungen auf dem Endgerät und der Information des Nutzers über Sicherheitsmaßnahmen. Ihre Gemeinsamkeit liegt darin, dass sie beide darauf abzielen, den Nutzer über die Bedeutung und Umsetzung von Sicherheitsvorkehrungen zu informieren, um die Sicherheit der Anwendung und der Daten auf dem Endgerät zu erhöhen.
Der Prüfaspekt O.Plat_1 (vgl. Anhang A.10) fordert die Prüfung, dass auf dem Endgerät ein aktiver Geräteschutz aktiviert ist. Die Einrichtung von beispielsweise Passwort, Mustersperre oder PIN zum Entsperren des Endgerätes und somit das Erlangen des Zugriffs auf das Endgerät sind als ausreichender Geräteschutz anzusehen. Um vor unbefugten Zugriff auf das Endgerät zu schützen, sollte der Geräteschutz aktiviert sein. Stellt die Anwendung bei der geforderten Prüfung fest, dass kein Geräteschutz aktiviert ist, muss der Nutzer darüber in Kenntnis gesetzt werden. Das kann in Form von einer Warnmeldung umgesetzt werden. Die Warnmeldung soll dem Nutzer die Aktivierung des

Geräteschutzes empfehlen und die verbundenen Risiken bei Nicht-Aktivierung erläutern, also dass unbefugte Zugriffe auf die Anwendung und die darauf gespeicherten Daten möglich sind.
Der Prüfaspekt O.Plat_16 (vgl. Anhang A.10) fordert ebenfalls, dass der Nutzer ausreichend in Kenntnis gesetzt wird. Allerdings ist dieser Prüfaspekt allgemeiner und breiter definiert. Denn der Prüfaspekt besagt, dass der Nutzer über Sicherheitsmaßnahmen informiert werden soll, die er selbst in den Einstellungen des Endgerätes oder in Anwendungen umsetzen kann, um die Sicherheit zu erhöhen. Welche weiteren Sicherheitsmaßnahmen das sein könnten, werden vom BSI nicht spezifiziert. Es folgt eine Auswahl möglicher Hinweise an Nutzern, mit deren Umsetzung dieser in der Lage ist, die Sicherheit zu erhöhen: Regelmäßige Aktualisierung des Betriebssystems, Aktualisierung von Anwendungen und Bibliotheken, Überprüfung der Anwendungsberechtigungen, Vermeidung öffentlicher WLAN-Hotspots, Bewusstsein für Phishing und Social Engineering schaffen. Diese Liste ist durch verschiedene weitere Sicherheitsmaßnahmen erweiterbar.

Fazit Die Umsetzung von O.Plat_1 könnte noch erweitert werden, indem die Warnmeldung direkt über eine Verlinkung zu der Aktivierung des Geräteschutzes verfügt, sodass Nutzern, die sich nicht ausreichend mit den Geräteeinstellungen auskennen, die Möglichkeit haben, die Einstellung zu aktivieren. Zu O.Plat_16 existieren seitens BSI keine expliziten Empfehlungen welche Sicherheitsmaßnahmen der Nutzer aktiv ergreifen kann, bis auf den Geräteschutz, der bereits durch O.Plat_1 abgedeckt worden ist. Auch an welcher Stelle und wie oft dieser Hinweis dem Nutzer präsentiert wird, ist nicht festgelegt worden. Dieser Prüfaspekt bietet also Interpretationsspielraum hinsichtlich der Umsetzung, weshalb genau abgewogen werden sollte, wie man ihn umsetzen möchte.

4.10.2 Prüfaspekte O.Plat_2 und O.Plat_3

Prüfaspekt O.Plat_2 (vgl. Anhang A.10) besagt, dass eine Anwendung nur die Berechtigungen einfordern darf, die für ihre grundlegende Funktionalität unbedingt erforderlich sind. Das bedeutet, dass die Anwendung keine unnötigen Berechtigungen anfordern sollte, die über ihren Hauptzweck hinausgehen. Hier ist ein Abgleich des primären Zwecks mithilfe der Prüfaspekte des Anwendungszwecks aus Abschnitt 4.1 durchzuführen. Berechtigungen können nicht ohne das Einverständnis des Nutzers freigegeben werden. Die angefragten Berechtigungen werden demnach dem Nutzer zu Freigabe dargelegt. Prüfaspekt O.Plat_3 (vgl. Anhang A.10) fordert, dass der Nutzer über den Zweck der an-

gefragten Berechtigung aufgeklärt werden muss. Die Erläuterungen der Berechtigungen sollten dem Nutzer in Form von Benachrichtigungen oder Dialogfeldern angezeigt werden, wenn die Berechtigungen das erste Mal angefordert werden. Außerdem sollen die Auswirkungen beschrieben werden, die eintreten, falls der Nutzer es wünscht, die Berechtigung nicht zu erteilen. Denn eine Nicht-Erteilung einer Berechtigung, die dem primären Zweck der Anwendung dient, könnte die Funktionalitäten oder die Nutzbarkeit der Anwendung gegebenenfalls stark einschränken.

Fazit Beide Prüfaspekte zielen es darauf ab, sicherzustellen, dass die Berechtigungen angemessen und transparent verwendet werden und der Nutzer über ihre Zwecke informiert wird. Dadurch soll die Privatsphäre und Sicherheit des Nutzers geschützt werden, indem unnötige Berechtigungen vermieden und der Nutzer in die Lage versetzt wird, informierte Entscheidungen zu treffen.

4.10.3 Prüfaspekte O.Plat_4 und O.Plat_5

In beiden Prüfaspekten geht es darum, wie die Anwendung mit sensiblen Daten in Meldungen und Benachrichtigungen umgehen soll. Beispiele für Meldungen und Benachrichtigungen sind: Push-Benachrichtigungen, Pop-up Fenster und Benachrichtigungsleiste. In solchen Benachrichtigungen und Meldungen dürfen keine sensiblen Daten enthalten sein (O.Plat_4, s. A.10). Prüfaspekt O.Plat_5 (vgl. Anhang A.10) bietet die Ausnahme, dass sensible Daten auch in Meldungen und Benachrichtigungen enthalten sein können, wenn der Nutzer diese explizit einschaltet. Die Anwendung kann diese Option bieten, sollte aber auch über die daraus resultierenden Risiken aufgeklärt werden. Die Option muss bei Werkseinstellung deaktiviert sein.

Fazit Die Ausnahme zur Darstellung von sensiblen Daten wurde zu wenig konkretisiert. Es ist unklar, ob die Anwendung konkret auf die Option des Einschaltens der Meldungen und Benachrichtigungen mit sensiblen Daten hinweisen darf. Auf jeden Fall sollte diese Option dem Nutzer nur empfohlen werden, wenn es den primären Zweck der Anwendung dient. Ansonsten sollten, um sicherzugehen, keine sensiblen Daten enthalten sein.

4.10.4 Prüfaspekte O.Plat_6 und O.Plat_7

O.Plat_6 (vgl. Anhang A.10) stellt die Anforderung, dass der Zugriff auf vorgesehene Dateipfade beschränkt wird. Die Anwendung sollte sicherstellen, dass nur bestimmte Dateipfade für den Zugriff durch die Anwendung freigegeben sind. Andere sensible Bereiche des Dateisystems sollten nicht zugänglich sein. Dies kann durch die Verwendung von Berechtigungen und Zugriffsregeln erfolgen, die durch das Betriebssystem oder die Anwendung selbst festgelegt werden.
O.Plat_5 (vgl. Anhang A.10) fordert die Realisierung von Zugriffsbeschränkungen auf sämtliche Daten. Die Anwendung muss sicherstellen, dass der Zugriff auf sensible Daten innerhalb der Anwendung angemessen eingeschränkt ist. Dies kann durch die Verwendung von Berechtigungen und Zugriffssteuerungen erfolgen, um sicherzustellen, dass nur autorisierte Teile der Anwendung oder bestimmte Benutzer (Rollen) auf bestimmte Daten zugreifen dürfen.

Fazit Beide Prüfaspekte betreffen den Zugriff auf Daten und Dateipfade in der Anwendung. Durch die Maßnahmen von Berechtigungen, Zugriffssteuerungen und Zugriffsbeschränkungen wird verhindert, dass unautorisierte Personen auf sensible Informationen zugreifen können, was das Risiko von Datenschutzverletzungen und Sicherheitsproblemen minimiert.

4.10.5 Prüfaspekte O.Plat_8, O.Plat_9 und O.Plat_10

O.Plat_8 und O.Plat_9 beziehen sich auf die Verwendung und Einschränkung von Broadcast-Nachrichten. Broadcast ist ein Mechanismus, bei dem Anwendungen oder Systeme Nachrichten senden oder empfangen können, um über bestimmte Ereignisse zu informieren. Es funktioniert ähnlich wie ein Publish-Subscribe-Designmuster, bei dem Sender (Publisher) Nachrichten an Empfänger (Subscriber) schicken, die sich für diese Art von Nachrichten interessieren [32]. Broadcasts ermöglichen somit die Kommunikation zwischen verschiedenen Komponenten oder Anwendungen.

Broadcast-Nachrichten müssen folgende Anforderungen erfüllen:

- Sie müssen auf autorisierte Anwendungen beschränkt werden (O.Plat_8, s. A.10)
- Sie dürfen keine sensiblen Daten enthalten (O.Plat_9, s. A.10)

O.Plat_10 (vgl. Anhang A.10) bezieht sich auf die Interprozesskommunikation (IPC). Interprozesskommunikation ist ein Konzept, das es verschiedenen Prozessen oder Anwendungen ermöglicht, miteinander zu kommunizieren und Informationen auszutauschen [4]. Das BSI formuliert die Anforderung, dass keine sensiblen Funktionalitäten über die Interprozesskommunikation angeboten werden dürfen. Mit Ausnahme, dass das Anbieten den primären Zweck der Anwendung erfüllt (siehe Abschnitt 4.1). Angebotene sensible Funktionalitäten müssen geschützt werden. Die Implementierung geeigneter Sicherheitsmechanismen, Verschlüsselungstechniken oder Authentifizierungsverfahren können dem Abschnitt 4.5 entnommen werden. Bei sensiblen Funktionalitäten handelt es sich grundsätzlich um alle Funktionen oder Aktionen, die den Zugriff auf vertrauliche Informationen ermöglichen oder die Integrität und Sicherheit der Anwendung beeinflussen könnten.

Fazit Diese Prüfaspekte betonen die Notwendigkeit, den Zugriff auf Broadcast-Nachrichten und IPC-Funktionen einzuschränken und sicherzustellen, dass sensible Daten nicht unnötig freigegeben werden.

4.10.6 Prüfaspekte O.Plat_11 und O.Plat_13

Diese Prüfaspekte beziehen sich auf die Anwendung von WebViews. WebViews sind Komponenten in mobilen Anwendungen, die es ermöglichen, Webinhalte innerhalb der Anwendung anzuzeigen, indem sie einen integrierten Webbrowser darstellen [20] [78]. Wenn die Anwendung keine WebViews anwendet, müssen diese Prüfaspekte nicht erfüllt werden.

WebView Inhalte bestehen in der Regel aus HTML, CSS und JavaScript. Der Prüfaspekt O.Plat_11 (vgl. Anhang A.10) fordert aber in erster Linie, dass die WebView-Komponenten Script-Sprachen deaktivieren sollen. Nur wenn Javascript unabdingbar ist, darf es unter der Voraussetzung verwendet werden, dass sich auf Quellen unter der Kontrolle des Herstellers beschränkt wird. Die Gefahr bei der Darstellung von Webinhalten mit JavaScript liegt unter anderem darin, dass bösartiger Javascript-Code ausgeführt werden könnte. Dieser könnte das sogenannte Sandbox Modell, das normalerweise von eigenständigen Browsern verwendet wird um Webinhalte voneinander und von anderen Teilen des Betriebssystems zu isolieren, umgehen und auf die sensiblen Daten auf dem Gerät zugreifen [78] [87] [38].

O.Plat_13 (vgl. Anhang A.10) fordert, dass alle nicht benötigten Protokoll-Handler deaktiviert werden sollen. Ein Protokoll-Handler in WebViews ist ein Mechanismus, der

festlegt, wie URLs mit bestimmten Protokollen (z. B. HTTP, HTTPS) in der WebView behandelt werden sollen. Die WebView-Komponente sollte nicht benötigte Protokoll-Handler deaktivieren. Zum Beispiel für den HTTP-Protokoll-Handler sollte festlegt werden, dass er die Verarbeitung von URLs mit dem HTTP-Protokoll ablehnt. Dadurch wird verhindert, dass die WebView URLs mit dem unsicheren HTTP-Protokoll lädt und somit nur URLs mit dem sicheren HTTPS-Protokoll akzeptiert.

Fazit Diese Prüfaspekte sind nur applikabel, wenn WebViews angewendet werden. Es existieren viele Forschungen, die die Sicherheit in Bezug auf JavaScript in WebViews analysieren und auf den Entschluss kommen, dass JavaScript ein Sicherheitsrisiko darstellt (siehe [78] [87] [38]). Aus diesem Grund sind die Bedenken des BSI hinsichtlich der Verwendung von JavaScript durchaus berechtigt, es schränkt aber die Entwicklung der WebView Inhalte ein, beziehungsweise erhöht den Aufwand der Entwickler, indem sie sich auf Quellen unter eigener Kontrolle beschränken müssen. Denn Webinhalte ohne JavaScript, also nur HTML und CSS, würden sich nur eignen, um Informationen darzustellen. Das könnte Anwendung für die Darstellung der Datenschutzerklärung finden.

4.10.7 Prüfaspekte O.Plat_12, O.Plat_14 und O.Plat_15

Diese Prüfaspekte beziehen sich auf das Verhalten der Anwendung im Hintergrundbetrieb und beim Beenden.
O.Plat_12 (vgl. Anhang A.10) besagt, dass wenn eine mobile Anwendung in den Hintergrundbetrieb wechselt (zum Beispiel wenn der Benutzer die Anwendung verlässt oder zu einer anderen Anwendung wechselt), müssen alle sensiblen Daten aus der aktuellen Ansicht der Anwendung entfernt werden. Dabei bezieht sich 'Ansicht' auf die momentan sichtbaren Bestandteile der Benutzeroberfläche wie 'Views' in iOS [55] oder 'Activities' in Android [31]. Das bedeutet, dass die Anwendung sicherstellen muss, dass keine vertraulichen Informationen in den sichtbaren Elementen verbleiben, um die Sicherheit der Daten zu gewährleisten, wenn die Anwendung im Hintergrund läuft oder vom Benutzer nicht aktiv genutzt wird. In Abschnitt 4.7.1 wurde bereits der Vorschlag vorgelegt, dass das Anzeigeelemente vorzugsweise geschwärzt werden soll bei Wechsel in den Hintergrundbetrieb, damit im App Switcher keine sensiblen Daten angezeigt werden. Somit wird sichergestellt, dass ungewollt sensible Informationen angezeigt werden.

O.Plat_14 und O.Plat_15 (vgl. Anhang A.10) hingegen konzentrieren sich auf Anforderungen an das Beenden der Anwendung. Nach dem Beenden der Anwendung müssen folgende Punkte erfüllt sein:

- Falls WebViews (siehe Abschnitt 4.10.6) verwendet werden, müssen anwendungsspezifische Cookies gelöscht werden.
- Nutzerspezifische Daten müssen sicher aus dem Arbeitsspeicher gelöscht, beziehungsweise überschrieben werden (siehe sicheres Löschen in Abschnitt 4.3.4). Es darf sich nicht nur auf das Betriebssystem oder den Garbage Collector verlassen werden, sondern eine aktive Aktion der Anwendung wird erwartet.

Fazit In O.Plat_12 spricht das BSI vom Entfernen der sensiblen Daten aus der aktuellen Ansicht der Anwendung. Es ist nicht sicher, was genau unter Entfernen gemeint ist. Es kann sein, dass der oben beschriebene Ansatz des Ausblendens der aktuellen Ansicht ausreichend ist oder das Löschen der Daten oder das Zurücksetzen der Ansicht auf einen sicheren Zustand gemeint ist. Diese Ansätze sind bei der Wahl der Umsetzung dieses Prüfaspektes zu berücksichtigen.

4.10.8 Bewertung Plattformspezifische Interaktionen

Bewertungsaspekt → Prüfaspekt ↓	Aufwand	Umsetz-barkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
O.Plat_1, O.Plat_16	mittel	mittel	ungenau	mittel
O.Plat_2, O.Plat_3	niedrig	leicht	eindeutig	niedrig
O.Plat_4, O.Plat_5	niedrig	leicht	mittel	niedrig
O.Plat_6, O.Plat_7	mittel	mittel	mittel	mittel
O.Plat_8, O.Plat_9, O.Plat_10	mittel	mittel	mittel	mittel
O.Plat_11, O.Plat_13	hoch	schwer	eindeutig	niedrig
O.Plat_12, O.Plat_14, O.Plat_15	mittel	mittel	ungenau	mittel
Gesamtbewertung	●	●	●	●

Tabelle 4.10: Bewertung Plattformspezifische Interaktionen

4.11 Resilienz

Dieses Kapitel bezieht sich auf die Prüfaspekte, die in die Kategorie Resilienz fallen. Die darin enthaltenen Prüfaspekte legen Anforderungen an die Widerstandsfähigkeit und Stabilität für DiGA fest.

4.11.1 Prüfaspekt O.Resi_1

Der Prüfaspekt O.Resi_1 (vgl. Anhang A.11) knüpft an den Prüfaspekt O.Plat_16 (4.10.1) an. Wie auch dort beschrieben, sollte der Nutzer über ihm mögliche Sicherheitsmaßnahmen informiert werden. O.Resi_1 bezieht sich aber primär auf den sicheren Umgang und der Konfiguration der Anwendung. Best-Practice Empfehlungen können folgende Informationen sein wie die Verwendung starker Passwörter, die Aktualisierung der Anwendung auf die neueste Version, der Umgang mit sensiblen Informationen, die Vermeidung von öffentlichem WLAN für vertrauliche Aktivitäten und der Schutz vor Social

Engineering-Angriffen. Die Anwendung sollte außerdem den Nutzer bei der Konfiguration unterstützen und sicherstellen, dass die Standardeinstellungen angemessen sicher sind. Weiterführend sollten dem Nutzer klare Warnungen angezeigt werden, wenn bestimmte Aktionen ein Sicherheitsrisiko darstellen könnten (siehe beispielsweise Abschnitt 4.11.2). Die jeweiligen möglichen Empfehlungen sind anwendungsspezifisch und können dadurch für jede Art der Anwendung unterschiedlich sein.

Fazit Die Aufklärung und das Schulen des Sicherheitsbewusstseins des Nutzers stehen in diesem Prüfaspekt im Vordergrund. Unabhängig davon, wie viele Sicherheitsmechanismen eine Anwendung implementiert, ist die Schwachstelle der Endnutzer, der die Sicherheit der Anwendung durch Unwissenheit beeinträchtigen kann. Aus diesem Grund ist es notwendig, dem Nutzer bewusst zu machen, was er selbst tun kann, um seine Daten zu schützen. Daher ist es auch wichtig, die möglichen Ansätze in einfacher Sprache zu formulieren und Beispiele zu geben sowie über resultierende Risiken zu informieren. Vorzugsweise sollten bestimmte Hinweise erst dann dem Nutzer angezeigt werden, wenn sie von Bedeutung sind, damit der Nutzer nicht mit zu vielen Informationen auf einmal konfrontiert wird.

4.11.2 Prüfaspekte O.Resi_2, O.Resi_3, O.Resi_4, O.Resi_5, O.Resi_6, O.Resi_7 und O.Resi_9

Diese Prüfaspekte zielen darauf ab, die Sicherheit und Integrität der Anwendung zu gewährleisten und potenzielle Bedrohungen zu erkennen. Sie schreiben Maßnahmen zum Schutz vor unterschiedlichen Angriffsszenarien und Sicherheitsbedrohungen vor.

Prüfaspekt O.Resi_2 (vgl. Anhang A.11) befasst sich mit der Root und Jailbreak Erkennung. Jailbreak und Root bezeichnen den Prozess, bei dem Nutzer erweiterte Zugriffsrechte auf einem mobilen Gerät (Jailbreak bei iOS, Root bei Android) erlangen, um tiefgreifende Änderungen am Betriebssystem vorzunehmen und zusätzliche Freiheiten zu erhalten, die normalerweise vom Hersteller oder Betriebssystemanbieter eingeschränkt sind [39] [70]. Es können also unautorisierte Änderungen am Betriebssystem vorgenommen werden. Bei iOS werden ermöglicht der Jailbreak die Ausführung von Code, der nicht von Apple genehmigt und signiert wurde, was dazu führt, dass die Installation zusätzlicher Anwendungen, Erweiterungen und Patches außerhalb des Apples App Store, möglich sind. Rooting bei Android versetzt den Nutzer in die Lage das Betriebssystem

des Geräts vollständig zu entfernen und ersetzen zu können [88].
Zusammenfassend können gerootete / gejailbreakte Geräte die Sicherheit des Gerätes beeinträchtigen und das Risiko von Sicherheitsverletzungen erhöhen. Dadurch, dass Betriebssystem spezifische Sicherheitsmechanismen überschrieben werden, stellen solche Geräte ein großes Sicherheitsrisiko dar.

Die Anwendung sollte eine Jailbreak / Root Detection implementieren. Es können verschiedene Techniken eingesetzt werden, um zu erkennen, ob das Gerät gerootet oder gejailbreakt wurde.

- **Überprüfung von Systemdateien, Root Binaries und App Packages**
  Die Anwendung kann nach Root Binaries suchen, die im Root- beziehungsweise Admin-Kontext ausgeführt werden können. Außerdem kann die Anwendung eine Liste von bekannten App Packages erstellen, die mit Rooting / Jailbreak in Verbindung stehen. Das Vorhandensein solcher Binaries und App Packages auf dem Gerät könnten auf ein manipuliertes Gerät hindeuten. Auch Systemdateien, die normalerweise durch Rooting oder Jailbreaking verändert werden, sollten überprüft werden. Veränderte oder fehlende Dateien können auf Jailbreak / Root hinweisen.

- **Auswertung von Signaturen**
  Die Anwendung kann die Signaturen von Systemkomponenten überprüfen, um festzustellen, ob diese von einem vertrauenswürdigen Anbieter signiert wurden. Abweichungen von den erwarteten Signaturen können auf ein manipuliertes Gerät hinweisen.

- **Abfragen von Root- oder Jailbreak-Status-Frameworks**
  Es gibt Frameworks oder APIs, die Informationen über den Root- oder Jailbreak-Status des Geräts bereitstellen. Die Anwendung kann diese APIs nutzen, um den aktuellen Status zu ermitteln.

- **Magisk oder Cydia Erkennung**
  Magisk ist ein Tool für Android, dass Root-Zugriff ermöglicht, ohne das Betriebssystem dauerhaft zu verändern. Und Cydia ermöglicht es iOS Nutzern nicht autorisierte Anwendungen zu installieren [46].

Nach der Erkennung müssen geeignete Maßnahmen stattfinden, das BSI bietet die Option, den Nutzer über das Risiko zu informieren oder die Fortsetzung der Anwendung zu unterbinden. Dadurch, dass manipulierte Geräte ein gravierendes Sicherheitsrisiko darstellen, sollte die Fortsetzung der Anwendung auf jeden Fall unterbunden werden. Vor

dem Beenden der Anwendung sollte dem Nutzer die Information zur Verfügung gestellt werden, dass ein Root / Jailbreak auf dem Gerät detektiert worden ist, welche Risiken damit verbunden sind und dass aus diesen Gründen die Anwendung nicht ausführbar ist.

Neben der Root / Jailbreak Detection muss die Anwendung auch erkennen, wenn sie in einer Entwicklungs- oder Debug-Umgebung gestartet wird (O.Resi_3, s. A.11). Auf einem mobilen Gerät kann eine Entwicklungs- oder Debug-Umgebung aktiviert werden, indem bestimmte Einstellungen oder Optionen auf dem Gerät geändert werden oder indem das Gerät mit einem Computer über USB verbunden wird und Debugging-Tools (ADB für Android und Xcode für iOS) verwendet werden. Bei Erkennung muss der Start der Anwendung unterbunden werden. Außerdem muss die Anwendung den Start abbrechen, wenn sie mit ungewöhnlichen Benutzerrechten gestartet wird (O.Resi_4, s. A.11), darunter fallen root und nobody. Während root den Superuser oder Administrator darstellt und somit uneingeschränkte Rechte besitzt, einschließlich des Zugriffs auf geschützte Systemdateien und -ressourcen, werden mit nobody Prozesse mit minimalen Berechtigungen ausgeführt, also mit sehr eingeschränkten Rechten und Berechtigungen [2].

Es ist wichtig, dass die Integrität des Endgeräts vor der Verarbeitung sensibler Daten geprüft wird (O.Resi_5, s. A.11). Neben den bereits genannten Aspekten bezüglich des Einsatzes von modifizierten Geräten (Root oder Jailbreak), Vorhandensein von verdächtigen Werkzeugen oder Anwendungen, auch die Aktualität der Betriebssystemversion. Die Anwendung sollte prüfen, ob das Betriebssystem des Geräts auf dem neuesten Stand ist und alle verfügbaren Sicherheitsupdates und Patches installiert wurden, mehr Informationen zum Thema Updates, können dem Abschnitt 4.2.6 entnommen werden. Hierbei ist zu beachten, dass der Prüfaspekt O.Resi_9 (vgl. Anhang A.11) fordert, dass die Zugriffskontrollmechanismen unterschiedliche Plattformen und Plattformversionen berücksichtigen müssen. Es darf sich also bei der Implementierung der Zugriffsmaßnahmen nicht allein auf das Betriebssystem verlassen werden.

Mittels O.Resi_6 (vgl. Anhang A.11) soll außerdem sichergestellt werden, dass die Anwendung vor dem Zugriff auf das Hintergrundsystem sicherstellt, dass dieses System authentisch ist. Das BSI verweist hierbei auf O.Ntwk_4, wobei auch O.Ntwk_5 diesem Prüfaspekt ähnelt. Die mögliche Umsetzung der Authentizitätsprüfung durch Zertifikate kann dem Abschnitt 4.9.2 entnommen werden, der die verwiesenen Prüfaspekte beinhaltet. Zusätzlich fordert der Prüfaspekt O.Resi_7 (vgl. Anhang A.11), dass die Anwendung

eine Integritätsprüfung vor jeder Verarbeitung sensibler Daten innerhalb des Programmablaufs durchführen soll. Das bedeutet, dass die Integrität der Anwendung und ihrer Komponenten regelmäßig überprüft werden sollte, nicht nur beim Programmstart, sondern auch vor jedem Vorgang, bei dem sensible Daten verarbeitet werden.

Fazit In Bezug auf O.Resi_2 ist es wichtig zu beachten, dass die Erkennung von Rooting und Jailbreaks nicht immer zuverlässig sind. Es ist möglich, dass neue Techniken eingesetzt, um die Erkennung zu umgehen. Daher sollten mehrere Erkennungsmethoden kombiniert werden, um die Sicherheit der Anwendung zu erhöhen. Es ist interessant, dass das BSI die Weiterführung der Anwendung nach einer Aufklärung des Nutzers als mögliches Vorgehen ansieht, jedoch der Start in Entwicklungs- oder Debug-Umgebungen oder mit ungewöhnlichen Nutzerrechten sofort unterbunden werden sollen. Alle diese Fälle stellen ein hohes Sicherheitsrisiko dar und sollten aus diesem Grund unterbunden werden.

4.11.3 Prüfaspekt O.Resi_8

Dieser Prüfaspekt (vgl. Anhang A.11) stellt die Anforderung, dass starke Maßnahmen gegen Reverse Engineering umgesetzt werden müssen. Reverse Engineering bezeichnet den Prozess der Analyse und Verstehens des Quellcodes oder der Binärdateien einer Anwendung, um ihre Funktionsweise oder Sicherheitslücken zu untersuchen [9].

Um eine Anwendung vor Reverse Engineering zu schützen, gibt es folgende Möglichkeiten. Eine Möglichkeit ist das Code Packing [76], hierbei werden die ursprünglichen ausführbaren Dateien der Anwendung versteckt oder in eine verschlüsselte oder verschleierte Form umgewandelt, sodass wir sie nicht einfach umkehren, verändern und neu verpacken können. Das Verbieten von Debug Modus oder die Prüfung auf ein gerootetes System wurden bereits in dem vorherigen Abschnitt 4.11.2 behandelt und gelten auch als Sicherheitsmaßnahme gegen Reverse Engineering [76].

Das BSI legt besonderen Wert auf Verschleierungsmaßnahmen, die Reverse Engineering erschweren. Es müssen 'sämtliche Strings, Dateinamen und interne Namen von Klassen und Methoden innerhalb der Anwendung, die einem Angreifer Hinweise auf den Programmablauf geben können, verschleiert werden' [58]. Dies kann mittels White-Box Kryptografie erfüllt werden, welche das Ziel verfolgt, kryptographische Schlüssel und Algorithmen zu schützen, indem die Schlüssel und Daten so verschlüsselt und gemischt

werden, dass sie auch bei Einsicht in den Quellcode schwer zu extrahieren sind [15] [111]. Eine weitere Verschleierungsmaßnahme ist die Code-Obfuscation, durch welche der Quellcode so transformiert wird, dass er für Menschen schwer lesbar und verstehbar wird. Dies erschwert es Angreifern, den Code zu analysieren und zu dekompilieren [76]).

Eine Auflistung der gängigen Reverse-Engineering Tools können den Quellen [6] und [9] entnommen werden. Es ist wichtig zu erkennen, über welche Möglichkeiten ein Angreifer verfügt und wie dieser vorgehen würde, damit diese Bereiche entsprechend gesichert werden können.

Fazit Die vollständige Verhinderung von Reverse Engineering nahezu unmöglich ist, da ein äußerst engagierter Angreifer mit ausreichend Ressourcen möglicherweise Wege findet, die Sicherheitsmaßnahmen zu umgehen. Daher ist es ratsam, mehrere Schutzschichten zu implementieren und die Sicherheitsmaßnahmen regelmäßig zu überprüfen und zu aktualisieren, um einen angemessenen Schutz zu gewährleisten.

4.11.4 Prüfaspekt O.Resi_10

Dieser Prüfaspekt (vgl. Anhang A.11) fordert, dass die Anwendung widerstandsfähig sein muss und auch bei auftretenden Störungen, unvorhergesehenen Ereignissen oder Fehlbedienung angemessen funktionieren sollte, ohne dass dies zu einem Absturz oder einer fehlerhaften Ausführung führt. Als auftretende Störungen werden beispielsweise Störungen in der Stromversorgung und der Internetverbindung genannt. Das Ziel ist es, eine stabile und zuverlässige Anwendung bereitzustellen, die auch in schwierigen oder fehlerhaften Umgebungen ordnungsgemäß arbeitet.

Folgende Maßnahmen können ergriffen werden, um die Robustheit der Anwendungen in Bezug auf Störungen zu gewährleisten [72]:

- Verwendung von Fallback-Strategien, um auf alternative Ressourcen oder Datenquellen zurückzugreifen, wenn die primären Ressourcen nicht verfügbar oder gestört sind.
- Implementierung von Wiederholungsmechanismen für Netzwerkanfragen, um bei vorübergehenden Netzwerkausfällen die Anfrage erneut zu versenden.

- Implementierung einer robusten Fehlerbehandlung, um auftretende Fehler oder Störungen zu erkennen und angemessen darauf zu reagieren. Das kann die Behandlung von Ausnahmen und Fehlern sowie die Anzeige von angemessenen Fehlermeldungen oder Protokolldaten beinhalten. Außerdem sollte sichergestellt werden, dass die Anwendung nach einem Fehler oder einer Störung ordnungsgemäß wiederhergestellt und fortgesetzt werden kann.
- In einigen Fällen kann es sinnvoll sein, einen Notfall-Modus in der Anwendung zu implementieren, der eine vereinfachte Funktionalität bietet, wenn bestimmte Störungen auftreten.

Fazit Durch die Implementierung von Mechanismen zur Fehlererkennung, -behandlung und -wiederherstellung kann die Anwendung auch bei unvorhergesehenen Ereignissen oder Störungen stabil und fehlerfrei funktionieren. Dies ermöglicht eine kontinuierliche und zuverlässige Nutzung der Anwendung und minimiert Ausfallzeiten.

4.11.5 Bewertung Resilienz

Bewertungsaspekt → Prüfaspekt ↓	**Aufwand**	**Umsetz-barkeit**	**Präzision der BSI Anforderung**	**Höhe der Priorität im Gesamtprojekt**
O.Resi_1	mittel	leicht	mittel	niedrig
O.Resi_2, O.Resi_3, O.Resi_4, O.Resi_5, O.Resi_6, O.Resi_7, O.Resi_9	hoch	schwer	ungenau	mittel
O.Resi_8	mittel	schwer	ungenau	mittel
O.Resi_10	mittel	mittel	ungenau	mittel
Gesamtbewertung	🟡	🔴	🔴	🟡

Tabelle 4.11: Bewertung Resilienz

5 Fazit

Dieses Kapitel bildet den Abschluss der Arbeit und stellt das Fazit vor. In dem Abschnitt 5.1 wird die Arbeit prägnant zusammengefasst. Der Abschnitt 5.2 stellt die Ergebnisse der Analyse und Bewertung der Prüfaspekte des BSI vor. Darunter fällt die Gesamtbewertung der Prüfaspekt-Kriterien und ein daraus resultierendes Fazit, dass sich auf die in Abschnitt 1.1 definierten Ziele dieser Arbeit bezieht. Dieser Abschnitt führt auch eine reflektierte Betrachtung dieser Arbeit in Bezug auf das Vorgehen und der Ergebnisse durch. Der Abschnitt 5.3 gibt einen Ausblick auf weitere mögliche Forschungsrichtungen, die auf den erarbeiteten Erkenntnissen dieser Arbeit aufbauen können.

5.1 Zusammenfassung der Arbeit

Es wurden die elf Prüfaspekt-Kategorien der TR-03161 [58] betrachtet. Innerhalb dieser Kategorien wurden Prüfaspekte, die einen ähnlichen Bereich abdecken oder dasselbe Ziel verfolgen, gruppiert. Jeder Prüfaspekt beziehungsweise jede Prüfaspekt-Gruppierung wurde vorgestellt, die Bedeutung analysiert und Möglichkeiten der Umsetzung vorgeschlagen. Bei der Umsetzung wurde zwischen zwingend notwendigen (nach BSI oder Stand der Technik) und optionalen Vorschlägen unterschieden. Anschließend erfolgte ein Fazit. Basierend auf dieser Analyse, den Umsetzungsmöglichkeiten und dem Fazit wurde der Prüfaspekt beziehungsweise jede Prüfaspekt-Gruppierung bewertet. Die einzelnen Bewertungen bildeten die Gesamtbewertung der Prüfaspekt-Kategorie. Eine Übersicht der Gesamtbewertungen für alle Prüfaspekt-Kategorien können dem Abschnitt 5.2 entnommen werden.

5.2 Bewertung der Ergebnisse und abschließendes Fazit

In der folgenden Tabelle 5.1 werden alle Gesamtbewertungen der elf Prüfaspekte dargestellt:

Bewertungsaspekt → Prüfaspekt-Kategorie ↓	Aufwand	Umsetzbarkeit	Präzision der BSI Anforderung	Höhe der Priorität im Gesamtprojekt
Anwendungszweck	🟡	🟡	🟢	🟡
Architektur	🟡	🟡	🟡	🟡
Quellcode	🟢	🟢	🟡	🟡
Drittanbieter-Software	🟡	🔴	🔴	🟡
Kryptografische Umsetzung	🔴	🟡	🔴	🔴
Authentifizierung	🟡	🟡	🟡	🔴
Datensicherheit	🔴	🟡	🔴	🟡
Kostenpflichtige Ressourcen	🟡	🟢	🟡	🟢
Netzwerkkommunikation	🔴	🟡	🟡	🔴
Plattformspezifische Interaktionen	🟡	🟡	🟡	🟡
Resilienz	🟡	🔴	🔴	🟡

Tabelle 5.1: Übersicht der Gesamtbewertung aller Prüfaspekt-Kategorien

Aus Sicht der Hersteller und deren Entwicklern sind für die Planung und Ressourcenverteilung vor allem der Aufwand und die Höhe der Priorität im Projekt ausschlaggebend. Es sollte also in Bezug auf den Aufwand und der Höhe der Priorität im Gesamtprojekt besonders die Prüfaspekte der Kryptografischen Umsetzung und die Netzwerkkommunikation berücksichtigt werden. Es sollte sowohl genügend Zeit als auch genügend Ressourcen eingeplant werden und es sollte möglichst früh in der Planungsphase sowie der Entwicklungsphase auf diese Prüfaspekte geachtet werden.
Ebenfalls früh eingeplant werden sollte der ebenfalls hoch priorisierte Prüfaspekt Authentifizierung, da die Erfüllung dieser Anforderung nicht nur dem Aspekt der Sicherheit,

sondern auch dem der Benutzerfreundlichkeit gerecht werden muss. Auch der Aufwand der Datensicherheit sollte nicht unterschätzt werden.
Neben dem Aufwand und der Höhe der Priorität ist auch die Umsetzbarkeit ein entscheidender Faktor in der Entwicklung einer DiGA. Diese spiegelt nicht nur wider, wie einfach die Umsetzung ist, sondern auch, ob es so wirklich umsetzbar ist. Die roten Bewertungen der Umsetzbarkeit sind an der Stelle rot, an denen auch die Präzision der BSI Formulierung rot markiert ist. Das hat den Grund, dass eine unpräzise Formulierung der Anforderung auch Auswirkungen auf die Umsetzbarkeit haben kann. Beispielsweise tätigt das BSI widersprüchliche Aussagen bei der Kategorie Drittanbieter-Software, weshalb die korrekte Umsetzung nicht realisiert werden kann. In Bezug auf die Kategorie Resilienz sind die Vorgaben des BSI zu ungenau und es ist schwer abzusehen, was erwartet wird. Dies erschwert ebenso die Umsetzbarkeit. Allerdings wird in diesem Fall die Umsetzbarkeit auch erschwert durch die Tatsache, dass es nach aktuellen Stand der Technik keine ausreichenden Mittel existieren, die ein bestimmtes Sicherheitsniveau garantieren können.
Für die roten Bewertungen der Präzision der BSI Anforderungen gilt für die Hersteller, dass sie zusätzlich zu den in dieser Arbeit durchgeführten Analysen und Empfehlungen der Umsetzung, noch weitere Recherchen durchführen müssen und diese Prüfaspekte besonders vorsichtig bearbeiten sollten. Dadurch, dass die Formulierungen zu diesen Anforderungen allgemein gehalten und somit auch ungenau sind, wird die Erwartungshaltung des BSI nicht deutlich.
Die Prüfaspekt-Kategorien die besondere Aufmerksamkeit benötigen, sind durch die roten Bewertungen in dem jeweiligen Bewertungsaspekt verdeutlicht worden. Eine herausstechende Kategorie ist die der Kryptografischen Umsetzung, deren Bewertungen bis auf die der Umsetzbarkeit rot sind. Diesem Prüfaspekt ist besondere Aufmerksamkeit zu schenken, da dieser auch das Fundament für andere Prüfaspekte in anderen Kategorien legt.

Letztenendes profitieren nicht nur Hersteller von dieser Bewertung, auch das BSI könnte basierend auf diesen Punkten und Empfehlungen ihre Anforderungen überarbeiten, um mehr Transparenz zu schaffen. Damit werden schlussendlich die Entwicklung und das technische Zulassungsverfahren von DiGA erleichtert, wodurch die Digitalisierung ebenfalls gefördert wird. Abschließend dient diese Gesamtbewertung dazu, dass Hersteller diese nutzen können, um ihre Prozesse und Ressourcen zu optimieren.

Dadurch, dass die Bewertung so gestaltet ist, dass nur die jeweilige Prüfaspekt-Kategorie betrachtet und bewertet wird, ohne andere Kategorien mit einzubeziehen, treten in man-

chen Fällen Redundanzen auf. Ein Beispiel hierfür ist der Prüfaspekt Netzwerkkommunikation, der Teile der Anforderungen der Kryptografischen Umsetzung wiedergibt. Durch die gewählte Art der Bewertung muss jeder dieser Prüfaspekte einzeln betrachtet werden. Dementsprechend ist bei beiden der Aufwand als hoch eingestuft. Tatsächlich ist es aber so, dass wenn der Aufwand der Kryptografischen Umsetzung erbracht worden ist, dieser in der Netzwerkkommunikation nicht mehr berücksichtigt werden müsste. Es würde sich also empfehlen, über eine Bewertung nachzudenken, in der die Prüfaspekt-Kategorien übergreifend bewertet werden.
Dieser Ansatz der neuen Bewertung könnte in einer anderen Form der Darstellung der Prüfaspekte allgemein Anwendung finden. Wie in Kapitel 3 beschrieben worden ist, fiel die Entscheidung auf eine an die Prüfaspekt-Kategorien des BSI gebundene Bearbeitung. Während der Analyse der einzelnen Prüfaspekte wurde deutlich, dass teilweise redundante, widersprüchliche oder einander ergänzende Prüfaspekte verschiedenen Kategorien angehören. Es würde also durchaus Sinn ergeben, die Kategorien aufzulösen und neue zu formen. Beispielsweise würde sich eine Kategorie Sensible Daten empfehlen. In dieser Kategorie würden dann alle Anforderungen stehen, die den Umgang mit den sensiblen Daten beschreiben. Obwohl die Kategorie Drittanbieter-Software existiert, sind weitere Anforderungen in anderen Kategorien zu finden. Diese könnten auch so strukturiert werden, dass sich alle Drittanbieter Anforderungen in einer Kategorie befinden.
Der Vorteil dieser vorgeschlagenen Strukturierung liegt darin, dass diese Form der Darstellung der Anforderungen Entwickler-freundlicher wäre, indem sie an den Entwicklungsprozess angelehnt ist. Außerdem würde der Aufwand reduziert werden, indem nicht alle Kategorien durchgeschaut werden müssten, in denen zu dem Thema noch weitere Anforderungen an anderer Stelle existieren.

Diese Arbeit hatte das Ziel, alle Prüfaspekte zu berücksichtigen, also ein Ansatz, der in die Breite ging. Ein anderer Ansatz wäre der in die Tiefe zu gehen. Zum Beispiel könnten insgesamt weniger Prüfaspekt-Kategorien in die Betrachtung einbezogen werden, um sich auf einzelne hoch-priorisierte fokussieren zu können. Dazu würden Kategorien wie zum Beispiel die Kryptografische Umsetzung oder die Authentifizierung zählen. Anstatt also nur die Prüfaspekte zu analysieren, könnte auch zusätzlich auf die Testcharakteristika eingegangen werden oder die Umsetzungsvorschläge mit einer jeweils konkreten Implementierung ergänzt werden.
Die in dieser Arbeit verwendete Strukturierung, welche sich an der des BSI orientiert, sowie die Bewertungen können als geeignete Grundlage für die Umstrukturierung dienen, da in den einzelnen Analysen bereits Bezüge hergestellt worden sind.

Zusammenfassend lässt sich feststellen, dass die Prüfaspekte in der Technischen Richtlinie TR-03161 [58] noch zu ungenau formuliert sind, sodass Entwickler sich nicht sicher sein können, dass ihre Umsetzung der Anforderungen einer Prüfung durch eine Prüfstelle sicher standhalten wird. Die in dieser Arbeit vorgestellten Möglichkeiten basieren auf der Analyse der Anforderungen des BSI und weiterführender Recherchen. Sie dienen lediglich als Anhaltspunkt für Hersteller, um zu verstehen, was erwartet wird und wie es umgesetzt werden könnte. Es werden aber dennoch weitere Recherchen der Hersteller vorausgesetzt. Besonders auffällig waren die redundanten Anforderungen und die Querverweise zwischen den Prüfaspekt-Kategorien. An dieser Stelle besteht auf jeden Fall noch Optimierungsbedarf. Die Prüfaspekte dienen der Entwicklung sicherer Anwendungen, die darauf ausgelegt sind, die sensiblen Daten der Nutzer bestmöglich nach aktuellem Stand der Technik zu schützen. Dabei wird leider an manchen Stellen (wie beispielsweise bei der Authentifizierung) außer Acht gelassen, dass eine weitere grundlegende Anforderung an eine Anwendung ist, dass sie so benutzerfreundlich gestaltet ist, damit sie von allen vorgesehenen Nutzern problemlos und intuitiv verwendet werden kann.
Ziel sollte es sein, so eindeutig wie möglich die Erwartungen an die Sicherheitsanforderungen an eine DiGA zu definieren und diese Anforderungen sinnvoll zu strukturieren, um die Entwicklung von DiGA zu erleichtern. Dabei sollte auch die Benutzerfreundlichkeit berücksichtigt werden. DiGA sind ein wichtiger Bestandteil der Digitalisierung im Gesundheitswesen und aus diesem Grund sollte der Prozess der Entwicklung und Zulassung optimiert werden. Es sollte für Hersteller attraktiv sein, den teils komplexen und langwierigen Prozess auf sich zu nehmen, um im Anschluss eine zertifizierte mobile Anwendung anbieten zu können.

5.3 Ausblick

Die vorliegende Arbeit hat sich intensiv mit der Analyse und Umsetzung der Prüfaspekte aus der TR-03161 für mobile Anwendungen auseinandergesetzt, diese bewertet und Verbesserungsvorschläge dargelegt. Unter Berücksichtigung der bisherigen Forschungsergebnisse ergeben sich mehrere spannende Perspektiven, die es wert sind, vertieft zu werden. Eine mögliche Anpassung der Struktur der Prüfaspekt-Kategorien nach den Bedürfnissen der Entwickler sowie die Fokussierung auf einzelne Prüfaspekte und dafür eine tiefere Analyse durchzuführen, wurden bereits in Abschnitt 5.2 skizziert.

Da nur die Prüfaspekte im Fokus dieser Arbeit lagen, wäre eine Analyse und Bewertung der Testcharakteristika analog zu dieser Arbeit ein spannendes Themengebiet. Der Schwerpunkt bei der Analyse der Testcharakteristika bestünde darin, die möglichen Tests vorzustellen, die durchgeführt werden, um die Erfüllung der Prüfaspekte zu prüfen. Zum einen würde dieser Ansatz den Prüfstellen helfen, ihr Abläufe zu verifizieren oder gar zu erweitern, zum anderen hätten Hersteller die Möglichkeit, vor dem Zulassungsverfahren diese Tests durchzuführen, um die Qualität ihrer Anwendung zu erhöhen.

Ein zusätzliches Themengebiet, welches auf den Erkenntnissen dieser Arbeit aufbauen könnte, wäre die aktive Demonstration einer Anwendung sowie das Präsentieren konkreter Implementierungsbeispiele. Bei diesem Ansatz wird der Fokus auf entweder Android oder iOS gelegt. Durch diesen Fokus können konkrete Implementierungen gemäß den in dieser Arbeit vorgestellten Umsetzungsmöglichkeiten, dargelegt werden. Dies würde die Möglichkeit bieten, praxisnahe Beispiele für die jeweilige Plattform anzubieten und passende Code-Beispiele und technische Umsetzungen zu zeigen.

Diese Arbeit deckt den *Teil 1: Mobile Anwendungen* der TR-03161 ab. Wie in dem Abschnitt 2.4 erwähnt, existieren noch *Teil 2: Web-Anwendungen* und *Teil 3: Hintergrundsysteme*. Für sowohl Web-Anwendungen als auch Hintergrundsysteme kann diese in dieser Arbeit vorgestellte Analyse und Bewertung der Prüfaspekte umgesetzt werden.

Abschließend lässt sich festhalten, dass die vorliegende Arbeit einen wichtigen Schritt in Richtung einer umfassenden Analyse und Umsetzung der Prüfaspekte gemäß der Technischen Richtlinie TR-03161 für mobile Anwendungen im Gesundheitswesen darstellt. Die gewonnenen Erkenntnisse legen den Grundstein für zukünftige Entwicklungen und Forschungen auf diesem Gebiet. Die Umsetzung der hier skizzierten Themengebiete könnte dazu beitragen, dass die Erfüllung der technischen Anforderungen, die an DiGA gestellt werden, für Hersteller vereinfacht werden und der Prozess der Prüfung transparenter wird.

DiGA sind ein wesentlicher Teil der Digitalisierung des Gesundheitswesens, denn sie können letztendlich zu einer verbesserten Gesundheitsversorgung und einer an die Patienten angepassten Behandlung führen.

Literaturverzeichnis

[1] : *Best Practices der mobilen App-Entwicklung*. 27. Januar 2023. – URL https://doit.software/de/blog/app-entwicklung

[2] 3.0RC1, Linux Standard Base PDA S.: *9.2. User Group Names*. – URL https://refspecs.linuxbase.org/LSB_3.0.0/LSB-PDA/LSB-PDA/usernames.html

[3] Acar, Yasemin ; Stransky, Christian ; Wermke, Dominik ; Weir, Charles ; Mazurek, Michelle L. ; Fahl, Sascha: Developers need support, too: A survey of security advice for software developers. In: *2017 IEEE Cybersecurity Development (SecDev)* IEEE (Veranst.), 2017, S. 22–26

[4] Ahmad, Zuhaib: *Inter-Process Communication in Android - Lessons Learnings*. 03. Mai 2019. – URL https://dev.to/xuhaibahmad/inter-process-communication-in-android-lessons-learnings-1p6n

[5] al., Roßnagel A. et: *AUDITOR-Schutzklassenkonzept*. 14. Februar 2019. – URL https://www.auditor-cert.de/wp-content/uploads/2019/03/Schutzklassenkonzept_v2_final.pdf

[6] Albakri, Ashwag ; Fatima, Huda ; Mohammed, Maram ; Ahmed, Aisha ; Ali, Aisha ; Ali, Asala ; Elzein, Nahla M.: Survey on reverse-engineering tools for android mobile devices. In: *Mathematical Problems in Engineering* 2022 (2022), S. 1–7

[7] Andriani, Ria ; Wijayanti, Stevi E. ; Wibowo, Ferry W.: Comparision of AES 128, 192 and 256 bit algorithm for encryption and description file. In: *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)* IEEE (Veranst.), 2018, S. 120–124

[8] Arfaoui, Ghada ; Bultel, Xavier ; Fouque, Pierre-Alain ; Nedelcu, Adina ; Onete, Cristina: The privacy of the TLS 1.3 protocol. In: *Cryptology ePrint Archive* (2019)

[9] Asher, Syeda W. ; Jan, Sadeeq ; Tsaramirsis, George ; Khan, Fazal Q. ; Khalil, Abdullah ; Obaidullah, Muhammad: Reverse Engineering of Mobile Banking Applications. In: *Comput. Syst. Sci. Eng.* 38 (2021), Nr. 3, S. 265–278

[10] Balebako, Rebecca ; Marsh, Abigail ; Lin, Jialiu ; Hong, Jason ; Cranor, Lorrie F.: The privacy and security behaviors of smartphone app developers. In: *Workshop on Usable Security* Citeseer (Veranst.), 2014, S. 1–10

[11] Ball, Alexander: Review of data management lifecycle models. (2012)

[12] Balzert, Helmut ; Balzert, Helmut: Authentifizierung und Autorisierung. In: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb* (2011), S. 153–175

[13] Baumann, Ulrike ; Franz, Elke ; Pfitzmann, Andreas ; Baumann, Ulrike ; Franz, Elke ; Pfitzmann, Andreas: Praktischer Betrieb kryptographischer Systeme. In: *Kryptographische Systeme* (2014), S. 107–127

[14] Benssalah, Mustapha ; Rhaskali, Yesser ; Drouiche, Karim: An efficient image encryption scheme for TMIS based on elliptic curve integrated encryption and linear cryptography. In: *Multimedia Tools and Applications* 80 (2021), Nr. 2, S. 2081–2107

[15] Beunardeau, Marc ; Connolly, Aisling ; Geraud, Remi ; Naccache, David: White-box cryptography: Security in an insecure environment. In: *IEEE Security & Privacy* 14 (2016), Nr. 5, S. 88–92

[16] Bourgeat, Thomas ; Lebedev, Ilia ; Wright, Andrew ; Zhang, Sizhuo ; Arvind ; Devadas, Srinivas: Mi6: Secure enclaves in a speculative out-of-order processor. In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, S. 42–56

[17] Bratan, Tanja ; Schneider, Diana ; Heyen, Nils B. ; Pullmann, Liliya ; Friedewald, Michael ; Kuhlmann, Dirk ; Brkic, Nicole ; Hüsing, Bärbel: E-Health in Deutschland: Entwicklungsperspektiven und internationaler Vergleich / Studien zum deutschen Innovationssystem. 2022. – Forschungsbericht

[18] Buhov, Damjan ; Huber, Markus ; Merzdovnik, Georg ; Weippl, Edgar: Pin it! Improving Android network security at runtime. In: *2016 IFIP Networking Conference (IFIP Networking) and Workshops* IEEE (Veranst.), 2016, S. 297–305

[19] Chen, Eric Y. ; Pei, Yutong ; Chen, Shuo ; Tian, Yuan ; Kotcher, Robert ; Tague, Patrick: Oauth demystified for mobile application developers. In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, S. 892–903

[20] Chin, Erika ; Wagner, David: Bifocals: Analyzing webview vulnerabilities in android applications. In: *Information Security Applications: 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers 14* Springer (Veranst.), 2014, S. 138–159

[21] Corporation, IBM: *Erstellen eines Java-Keystore und einer Zertifikatssignieranforderung zur Vorlage bei einer Zertifizierungsstelle.* April 2021

[22] Craig Jackson, Scott R. ; Sons, Susan: *Security from First Principles.* Oktober 2017

[23] Crespo-Martínez, Ignacio S. ; Campazas-Vega, Adrián ; Manuel Guerrero-Higueras Ángel ; Riego-DelCastillo, Virginia ; Aparicio, Claudia Álvarez ; Fernández-Llamas, Camino: SQL injection attack detection in network flow data. In: *Computers Security* (2023)

[24] Dahal, Ram K. ; Bhatta, Jagdish ; Dhamala, Tanka N.: Performance analysis of SHA-2 and SHA-3 finalists. In: *International Journal on Cryptography and Information Security (IJCIS)* 3 (2013), Nr. 3, S. 720–730

[25] Datenschutzausschuss, Europäischer: *Leitlinien 05/2020 zur Einwilligung gemäß Verordnung 2016/679.* edpb. 4. Mai 2020

[26] Developers, Google: *Informationen für den Abschnitt zur Datensicherheit von Google Play angeben.* – URL https://support.google.com/googleplay/android-developer/answer/10787469?sjid=17372748892758614041-EU#zippy=%2Cdata-types

[27] Developers, Google: *Security for Android Developers.* Januar 2020. – URL https://developer.android.com/quality/privacy-and-security

[28] Developers, Google: *BiometricPrompt*. July 2021. – URL https://developer.android.com/reference/androidx/biometric/BiometricPrompt

[29] Developers, Google: *Android Keystore system*. 07. Oktober 2022. – URL https://developer.android.com/training/articles/keystore

[30] Developers, Google: *Network security configuration*. 21. September 2022. – URL https://developer.android.com/training/articles/security-config

[31] Developers, Google: *Activity*. 07. Juni 2023. – URL https://developer.android.com/reference/android/app/Activity

[32] Developers, Google: *Broadcasts overview*. 26. Juli 2023. – URL https://developer.android.com/guide/components/broadcasts

[33] Dowling, Benjamin ; Fischlin, Marc ; Günther, Felix ; Stebila, Douglas: A cryptographic analysis of the TLS 1.3 handshake protocol. In: *Journal of Cryptology* 34 (2021), Nr. 4, S. 37

[34] DSK: *Kurzpapier Nr. 20 Einwilligung nach der DS-GVO*. 22. Februar 2019

[35] Dönisch, Annette: *Herstellerverbände kritisieren neue BSI-Richtlinie*. 15. Juni 2022. – URL https://www.handelsblatt.com/inside/digital_health/gesundheitsdaten-herstellerverbaende-kritisieren-neue-bsi-richtlinie/28429674.html

[36] Eckhardt, Jens: *Widerruf und Datenlöschung gemäß DSGVO*. 05. November 2018

[37] Ekberg, Jan-Erik ; Kostiainen, Kari ; Asokan, Nadarajah: Trusted execution environments on mobile devices. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, S. 1497–1498

[38] El-Zawawy, Mohamed A. ; Losiouk, Eleonora ; Conti, Mauro: Vulnerabilities in Android webview objects: Still not the end! In: *Computers & Security* 109 (2021), S. 102395

[39] Geist, Dana ; Nigmatullin, Marat ; Bierens, Roel: Jailbreak/root detection evasion study on iOS and Android. In: *MSc System and Network Engineering* (2016)

[40] Gesundheitsversorgung, Spitzenverband D.: *Im Fokus: Regulatorische Anforderungen an DiGA*. 28. September 2022. – URL https://digitalversorgt.de/wp-content/uploads/2022/09/Im-Fokus-Regulatorische-Anforderungen-DiGA.pdf

[41] Gesundheitsversorgung, Spitzenverband D.: *Neues Gesetz sorgt für Überregulierung statt Beschleunigung für DiGA*. 14. Juli 2023. – URL https://digitalversorgt.de/wp-content/uploads/2023/07/DigiG-DiGA-II.pdf

[42] GmbH ePrivacy: *ePrivacyseal - Kriterienkatalog 3.0*. 20. Mai 2022. – URL https://www.eprivacy.eu/fileadmin/Redakteur/PDF/Kriterienkataloge/EPS_EU_Kriterienkatalog_Mai_2022.pdf

[43] Gomez-Barrero, Marta ; Kolberg, Jascha ; Busch, Christoph: Erkennung von Präsentationsangriffen auf Fingerabdruck Systemen: Aktueller Stand und Herausforderungen. In: *Datenschutz und Datensicherheit-DuD* 44 (2020), Nr. 1, S. 26–31

[44] H. Krawczyk, R. C.: *HMAC: Keyed-Hashing for Message Authentication*. Februar 1997. – URL https://datatracker.ietf.org/doc/html/rfc2104

[45] H. Krawczyk, R. C.: *HMAC: Keyed-Hashing for Message Authentication*. Februar 1997. – URL https://datatracker.ietf.org/doc/html/rfc2104

[46] Haupert, Vincent: *Sicherheit mobiler Bankgeschäfte zwischen Innovation und Regulierung*, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Dissertation, 2019

[47] Hühnlein, Detlef: Identitätsmanagement: Eine visualisierte Begriffsbestimmung. In: *Datenschutz und Datensicherheit-DuD* 32 (2008), Nr. 3, S. 161–163

[48] Inc., Apple: *App privacy details on the App Store*. – URL https://developer.apple.com/app-store/app-privacy-details/

[49] Inc., Apple: *HOW TO CODE SIGN IOS APPS*. – URL https://ioscodesigning.io/

[50] Inc., Apple: *iOS Security Framework*. – URL https://developer.apple.com/documentation/security

[51] Inc., Apple: *Keychain Services*. – URL https://developer.apple.com/documentation/security/keychain_services/

[52] Inc., Apple: *Local Authentication.* – URL https://developer.apple.com/documentation/localauthentication/

[53] Inc., Apple: *NSAppTransportSecurity.* – URL https://developer.apple.com/documentation/bundleresources/information_property_list/nsapptransportsecurity/

[54] Inc., Apple: *Prozess der App-Codesignierung in iOS und iPadOS.* – URL https://support.apple.com/de-de/guide/security/sec7c917bf14/web

[55] Inc., Apple: *Views and controls.* – URL https://developer.apple.com/documentation/uikit/views_and_controls

[56] Informationstechnik, Bundesamt für Sicherheit in der: *Leitfaden zur Entwicklung sicherer Webanwendungen.* – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Webanwendungen/Webanw_Auftragnehmer.pdf?__blob=publicationFile&v=1

[57] Informationstechnik, Bundesamt für Sicherheit in der: *Sichere Passwörter erstellen.* – URL https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Accountschutz/Sichere-Passwoerter-erstellen/sichere-passwoerter-erstellen_node.html

[58] Informationstechnik, Bundesamt für Sicherheit in der: *Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen - Teil 1: Mobile Anwendungen.* – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03161/BSI-TR-03161-1.pdf?__blob=publicationFile&v=11. – Version 2.0

[59] Informationstechnik, Bundesamt für Sicherheit in der: *Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen - Teil 2: Web-Anwendungen.* – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03161/BSI-TR-03161-2.pdf?__blob=publicationFile&v=8. – Version 1.0

[60] Informationstechnik, Bundesamt für Sicherheit in der: *Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen - Teil 3: Hintergrundsysteme.* – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03161/BSI-TR-03161-3.pdf?__blob=publicationFile&v=7. – Version 1.0

[61] Informationstechnik, Bundesamt für Sicherheit in der: *Zwei-Faktor-Authentisierung.* – URL https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Accountschutz/Zwei-Faktor-Authentisierung/zwei-faktor-authentisierung_node.html

[62] Informationstechnik, Bundesamt für Sicherheit in der: *Kryptographische Vorgaben fur Projekte der Bundesregierung Teil 4: Kommunikationsverfahren in Anwendungen.* 10. Januar 2020. – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03116/BSI-TR-03116-4.pdf?__blob=publicationFile&v=1

[63] Informationstechnik, Bundesamt für Sicherheit in der: *Kryptographische Verfahren: Empfehlungen und Schlusselllängen Teil 2 – Verwendung von Transport Layer Security (TLS) (TR-02102-2).* Januar 2021. – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=2

[64] Informationstechnik, Bundesamt für Sicherheit in der: *Kryptographische Verfahren: Empfehlungen und Schlusselllängen (TR-02102-1).* Januar 2021. – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=10

[65] IT, TÜV: *DiGAV: Zertifizierung und Pentests für DiGA-Hersteller ab sofort Pflicht.* – URL https://www.tuvit.de/de/aktuelles/pressemitteilungen/pressemitteilungen-detail/article/digav-zertifizierung-und-pentests-fuer-diga-hersteller-ab-sofort-pflicht/

[66] IT, TÜV: *Prüfung Zertifizierung nach BSI TR-03161 – Anforderungen an Gesundheitsanwendungen.* – URL https://www.tuvit.de/de/leistungen/cyber-security/bsi-tr-03161/

[67] Jararweh, Yaser ; Tawalbeh, Hala ; Moh'd, Abidalrahman u. a.: Hardware performance evaluation of SHA-3 candidate algorithms. (2012)

[68] Justiz, Bundesministerium für: *Verordnung über das Verfahren und die Anforderungen zur Prüfung der Erstattungsfähigkeit digitaler Gesundheitsanwendungen in der gesetzlichen Krankenversicherung (Digitale Gesundheitsanwendungen-Verordnung - DiGAV).* 20. Dezember 2022. – URL https://www.gesetze-im-internet.de/digav/BJNR076800020.html

[69] Jürgen Wolf, René K. und: *C von A bis Z.* Rheinwerk Computing, 2020. – URL https://s3-eu-west-1.amazonaws.com/gxmedia.galileo-press.de/leseproben/5701/leseprobe_rheinwerk_c_von_a_bis_z-das_umfassende_handbuch.pdf

[70] Kellner, Ansgar ; Horlboge, Micha ; Rieck, Konrad ; Wressnegger, Christian: False sense of security: A study on the effectivity of jailbreak detection in banking apps. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* IEEE (Veranst.), 2019, S. 1–14

[71] Klima, Vlastimil ; Gligoroski, Danilo: Generic collision attacks on narrow-pipe hash functions faster than birthday paradox, applicable to MDx, SHA-1, SHA-2, and SHA-3 narrow-pipe candidates. In: *Cryptology ePrint Archive* (2010)

[72] Kolbcom: *Art. 32 DSGVO.* – URL https://kolbcom.de/art-32-dsgvo/#:~:text=Art.-,32%20DSGVO,der%20Alltags%2D%20und%20Unternehmensprozesse.

[73] Kolberg, Jascha: Sicherheit und Datenschutz für Biometrische Systeme. In: *D22* (2022)

[74] Kretschmer, Michael: *Der Datenlebenszyklus als Sicherheitsmodell.* 23. September 2022

[75] Lenze, Burkhard ; Lenze, Burkhard: Asymmetrische Verschlüsselungsverfahren. In: *Basiswissen Angewandte Mathematik–Numerik, Grafik, Kryptik: Eine Einführung mit Aufgaben, Lösungen, Selbsttests und interaktivem Online-Tool* (2020), S. 263–294

[76] Liao, Yibin: *System techniques for reverse engineering mobile applications*, University of Georgia, Dissertation, 2018

[77] Lu, Hongqian K.: Keeping your API keys in a safe. In: *2014 IEEE 7th International Conference on Cloud Computing* IEEE (Veranst.), 2014, S. 962–965

[78] Luo, Tongbo ; Hao, Hao ; Du, Wenliang ; Wang, Yifei ; Yin, Heng: Attacks on WebView in the Android system. In: *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011, S. 343–352

[79] Lyko, Peter: Sicheres Löschen auf mobilen Endgeräten unter Android. (2013)

[80] Malkapuram, Sumana: *Best Practices for Application Session Management*. April 20 2023. – URL https://auth0.com/blog/application-session-management-best-practices/

[81] Marco-Gisbert, Hector ; Ripoll Ripoll, Ismael: Address space layout randomization next generation. In: *Applied Sciences* (2019)

[82] Medizinprodukte (BfArM), Bundesinsitut für Arzneimittel und: *Die Ausfüllhilfe zum Antragsportal zur Aufnahme in das DiGA-Verzeichnis nach § 139e SGB V*. 16. Juni 2020. – URL https://bfarm.de/SharedDocs/Downloads/DE/Medizinprodukte/diga_ausfuellhilfe.pdf;jsessionid=D7BDCD6877B089338FC98AE6AF1AC3C1.intranet242?__blob=publicationFile. – Version 1.0

[83] Medizinprodukte (BfArM), Bundesinsitut für Arzneimittel und: *Das Fast-Track-Verfahren fur digitale Gesundheitsanwendungen (DiGA) nach § 139e SGB V*. 13. April 2023. – URL https://www.bfarm.de/SharedDocs/Downloads/DE/Medizinprodukte/diga_leitfaden.pdf?__blob=publicationFile. – Version 3.2

[84] Meffert, Klaus: *Anonymisierung von Daten und Datenschutz: Was bedeutet das und welche Rechtsgrundlagen sind relevant?* 24. April 2023

[85] Milanov, Evgeny: The RSA algorithm. In: *RSA laboratories* (2009), S. 1–11

[86] M'Raihi, David ; Machani, Salah ; Pei, Mingliang ; Rydell, Johan: Totp: Time-based one-time password algorithm. 2011. – Forschungsbericht

[87] Neugschwandtner, Matthias ; Lindorfer, Martina ; Platzer, Christian: A View to a Kill:{WebView} Exploitation. In: *6th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 13)*, 2013

[88] OWASP: *Owasp mobile Security Project: Dangers of Jailbreaking Rooting*. 05. Januar 2016

[89] Pollicove, Matt: *Der ultimative Leitfaden zur tokenbasierten Authentifizierung*. 14. Februar 2023. – URL https://www.okta.com/de/identity-101/what-is-token-based-authentication/

[90] Qayyum, Bushra ; Qahtani, Dhaii al ; Naeem, Bushra ; Ali, Kamran ; Raza, Mohsin ; Rehman, Masood U.: Awareness of Kill Switch Application Among Mobile Phone Users. In: *2019 UK/China Emerging Technologies (UCET)* IEEE (Veranst.), 2019, S. 1–5

[91] Reddy, Lakshmi Eswari P. ; Nelaturu, Sarat Chandra B.: New Techniques for Protection of IoT Devices from Malicious Behavior Using Working Set Based System Call Whitelisting and Argument Clustering. In: *Journal of Algebraic Statistics* 13 (2022), Nr. 1, S. 178–186

[92] Rodríguez, Germán E. ; Torres, Jenny G. ; Flores, Pamela ; Benavides, Diego E.: Cross-site scripting (XSS) attacks and mitigation: A survey. In: *Computer Networks* (2020)

[93] Roos, Björn ; Baier, Harald ; Lyko, Peter: Zum sicheren Löschen auf Smartphones am Beispiel von Android. (2014)

[94] Schläpfer, Tobias ; Rüst, Andreas: Security on IoT devices with secure elements. In: *Embedded World Conference, Nuremberg, Germany, 26-28 Februar 2019* WEKA (Veranst.), 2019

[95] Series, OWASP Cheat S.: *Cross Site Scripting Prevention Cheat Sheet*. – URL https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

[96] Shehab, Mohamed ; Mohsen, Fadi: Securing OAuth implementations in smart phones. In: *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, 2014, S. 167–170

[97] Sikder, Ratul ; Khan, Md S. ; Hossain, Md S. ; Khan, Wazir Z.: A survey on android security: development and deployment hindrance and best practices. In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 18 (2020), Nr. 1, S. 485–499

[98] Sinaeepourfard, Amir ; Masip-Bruin, Xavier ; Garcia, Jordi ; Marín-Tordera, Eva: A survey on data lifecycle models: Discussions toward the 6vs challenges. In: *Technical Report (UPC-DAC-RR-2015–18)* (2015)

[99] Standardization, International O. for: *BSI Technical Guideline TR-03166 - Technical Guideline for Biometric Authentication Components in Devices for Authentication.* 23. Juni 2016. – URL https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03166/BSI-TR-03166.pdf?__blob=publicationFile&v=4

[100] Standardization, International O. for: *ISO/IEC 30107-1 Information technology — Biometric presentation attack detection —Part 1: Framework,“ International Organization for Standardization.* 2016

[101] Standards, National I. of ; Technology: *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication.* 06. Oktober 2016. – URL https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-38b.pdf

[102] Standards, National I. of ; Technology: *Recommendation for Key Management.* Mai 2020. – URL https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

[103] Švenda, Petr: Basic comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS). In: *URL https://www.fi.muni.cz/ xsvenda/docs/AE$_c$omparison$_i$pics04.pdf* 35(2016)

[104] Swierczynski, Pawel ; Leander, Gregor ; Paar, Christof: Keccak und der SHA-2. In: *Datenschutz und Datensicherheit-DuD* 37 (2013), Nr. 11, S. 712–719

[105] Tsolkas, Alexander ; Schmidt, Klaus ; Tsolkas, Alexander ; Schmidt, Klaus: Rollenkonzept. In: *Rollen und Berechtigungskonzepte: Identity-und Access-Management im Unternehmen* (2017), S. 41–66

[106] Uymatiao, Mariano Luis T. ; Yu, William Emmanuel S.: Time-based OTP authentication via secure tunnel (TOAST): A mobile TOTP scheme using TLS seed

exchange and encrypted offline keystore. In: *2014 4th IEEE International Conference on Information Science and Technology* IEEE (Veranst.), 2014, S. 225–229

[107] Wagner, Daniel: Patient Education–Apps für die Patientenkommunikation. In: *OP-JOURNAL* 37 (2021), Nr. 01, S. 24–27

[108] Wagner, Daniel: *Patient Education – Apps für die Patientenkommunikation.* Georg Thieme Verlag KG. Februar 2021

[109] Wagner, Klaus-P ; Hüttl, Thomas ; Backin, Dieter ; Vieweg, Iris ; Werner, Christian ; Wagner, Klaus-P ; Hüttl, Thomas ; Backin, Dieter: Projekte: Wie Informationssysteme erfolgreich eingeführt werden. In: *Einführung Wirtschaftsinformatik: IT-Grundwissen für Studium und Praxis* (2012), S. 217–236

[110] Werner Schindler, Bundesamt für Sicherheit in der I. Matthias Peter und: *KA Proposal for Functionality Classes for Random Number Generators.* 02. September 2022. – URL https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e.pdf?__blob=publicationFile&v=4

[111] Wiener, Michael J.: Applying software protection to white-box cryptography. In: *Proceedings of the 5th Program Protection and Reverse Engineering Workshop*, 2015, S. 1–2

[112] Wing, Jeannette M.: The data life cycle. In: *Harvard Data Science Review* 1 (2019), Nr. 1, S. 6

[113] Winkler, Fredrik: *Benutzerdefinierte Werttypen in C++*, Diplomarbeit. Universität Hamburg, Arbeitsbereich Softwaretechnik, 08.09, Dissertation, 2010

[114] Wissik, Tanja ; Ďurčo, Matej: Research data workflows: from research data lifecycle models to institutional solutions. In: *Selected papers from the CLARIN annual conference*, 2015, S. 94–107

[115] Wolf, Armin H. ; Leppla, Cindy: Harmonisierung von Datenlebenszyklus-Modellen. (2020)

A Prüfaspekte für Anwendungen im Gesundheitswesen

A.1 Prüfaspekt (1): Anwendungszweck

O.Purp_1 Der Hersteller MUSS die rechtmäßigen Zwecke der Anwendung und die Verwendung von personenbezogenen Daten vor der Installation offenlegen (etwa in der Beschreibung des App-Stores) und den Nutzer mindestens bei der erstmaligen Inbetriebnahme darüber informieren.

O.Purp_2 Die Anwendung DARF KEINE Daten erheben und verarbeiten, die nicht dem rechtmäßigen Zweck der Anwendung dienen.

O.Purp_3 Die Anwendung MUSS vor jeglicher Erfassung oder Verarbeitung personenbezogener Daten eine aktive und eindeutige Einwilligungserklärung des Nutzers einholen.

O.Purp_4 Daten, deren Verwendung der Nutzer nicht ausdrücklich zugestimmt hat, DÜRFEN NICHT von der Anwendung oder dem Hintergrundsystem erfasst oder genutzt werden.

O.Purp_5 Die Anwendung MUSS ermöglichen, dass der Nutzer eine bereits erteilte Einwilligung wieder entziehen kann. Der Nutzer MUSS vor der Einwilligung über die Möglichkeit des Widerrufs und die sich daraus ergebenden Veränderungen im Verhalten der Anwendung informiert werden.

O.Purp_6 Der Hersteller MUSS ein Verzeichnis führen, welches erkennen lässt, welche Nutzereinwilligungen vorliegen. Der nutzerspezifische Teil des Verzeichnisses MUSS für den Nutzer automatisiert einsehbar sein. Es SOLL eine Historie dieses Verzeichnisses angefordert werden können.

O.Purp_7 Setzt die Anwendung Frameworks und Bibliotheken von Dritten ein, SOLLEN alle verwendeten Funktionen für die rechtmäßigen Zwecke der Anwendung erforderlich sein. Die Anwendung SOLL anderweitige Funktionen sicher deaktivieren. Im Falle einer geringen Nutzung SOLL abgewogen werden, ob die Nutzung im Verhältnis zur Vergrößerung der Angriffsoberfläche durch die verwendeten Frameworks und Bibliotheken steht.

O.Purp_8 Sofern es nicht für den vorgesehenen primären Zweck einer Anwendung erforderlich ist, DÜRFEN sensible Daten NICHT mit Dritten geteilt werden. Dies betrifft auch die Ablage dieser Daten im öffentlichen Teil des Dateisystems. Die Anwendung MUSS den Nutzer über die Konsequenzen einer eventuellen Weitergabe von Anwendungsdaten vollumfänglich informieren und sein Einverständnis einholen (OPT-IN).

O.Purp_9 Die Anwendung DARF sensible Daten NICHT auf dem Bildschirm darstellen, außer dies ist für den primären Zweck der Anwendung erforderlich.

A.2 Prüfaspekt (2): Architektur

O.Arch_1 „Security“ MUSS ein fester Bestandteil des Softwareentwicklungs- und Lebenszyklus‘ für die gesamte Anwendung sein (vgl. „iOS Security Framework” [iOSSF], beziehungsweise „Security for Android Developers“ [SfAD]).

O.Arch_2 Bereits in der Designphase der Anwendung MUSS berücksichtigt werden, dass die Anwendung in der Produktivphase sensible Daten verarbeiten wird. Die Architektur der Anwendung MUSS dafür die sichere Erhebung, Verarbeitung, Speicherung und Löschung der sensiblen Daten in einem Datenlebenszyklus abbilden.

O.Arch_3 Der Lebenszyklus von kryptographischem Schlüsselmaterial MUSS einer ausgearbeiteten Richtlinie folgen, die Eigenschaften wie die Zufallszahlenquelle, detaillierte Angaben zur Aufgabentrennung von Schlüsseln, Ablauf von Schlüsselzertifikaten, Integritätssicherung durch Hash-Algorithmen etc., umfasst. Die Richtlinie SOLL auf anerkannten Standards wie [TR02102-2] und [NIST80057] basieren.

O.Arch_4 Backups und Cloud-Backups DÜRFEN KEINE sensiblen Daten im Klartext beinhalten.

O.Arch_5 Sicherheitsfunktionen MÜSSEN immer auf allen Außenschnittstellen und API- Endpunkten implementiert werden.

O.Arch_6 Die Anwendung MUSS einen Authentizitäts- und Integritätsschutz für die Anwendung und ihre Konfiguration gewährleisten. Die Anwendung SOLL dabei regelmäßig eine eigene Authentizitäts- und Integritätsprüfung des Anwendungs-Binaries, basierend auf einer digitalen Signatur mit Zertifikat, durchführen.

O.Arch_7 Nutzt die Anwendung Frameworks oder Bibliotheken von Dritten (etwa für Objektserialisierung), MUSS der Hersteller dem Nutzer Informationen über den Nutzungsumfang und die eingesetzten Sicherheitsmechanismen klar darstellen. Die Anwendung MUSS sicherstellen, dass diese Funktionen in sicherer Weise genutzt werden. Die Anwendung MUSS darüber hinaus sicherstellen, dass ungenutzte Funktionen durch Dritte nicht aktiviert werden können.

O.Arch_8 Interpretierter Code (Nicht gemeint ist Code der plattformspezifischen Programmiersprachen (z. B. Java oder Kotlin bei Android)), der in möglichen Interaktionen mit Benutzereingaben steht (WebViews mit JavaScript), DARF KEINEN Zugriff auf verschlüsselte Speicher oder Nutzerdaten haben, sofern es für die Erfüllung des primären Zwecks der Anwendung nicht zwingend erforderlich ist.

O.Arch_9 Der Hersteller MUSS dem Nutzer eine barrierearme Möglichkeit bereitstellen, um Sicherheitsprobleme zu melden. Die Kommunikation SOLL über einen verschlüsselten Kanal stattfinden.

O.Arch_10 Die Anwendung SOLL beim Start auf verfügbare sicherheitsrelevante Updates prüfen. Wenn ein sicherheitsrelevantes Update verfügbar ist, DARF die Anwendung sensible Daten NICHT mehr verarbeiten, ohne dieses Update einzuspielen. Der Nutzer MUSS über die Möglichkeit eines Updates und über ein durchgeführtes Update informiert werden.

O.Arch_11 Der Hersteller KANN die Anwendung und Updates über einen vertrauenswürdigen Kanal mit einem eigenen App-Store bereitstellen.

O.Arch_12 Werden die Anwendung und Updates nicht über die normalen Mechanismen des App-Stores der Hardwareplattform eingespielt, MÜSSEN die Authentizität der Datenquelle sowie des Updates vor dessen Anwendung durch kryptographische Maßnahmen positiv bestätigt werden.

A.3 Prüfaspekt (3): Quellcode

O.Source_1 Die Anwendung MUSS Eingaben aus nicht vertrauenswürdigen Quellen vor deren Verwendung prüfen, um potenziell bösartige Werte vor der Verarbeitung herauszufiltern.

O.Source_2 Die Anwendung MUSS eingehende und ausgehende Daten maskieren beziehungsweise von potenziell schadhaften Zeichen bereinigen oder deren Verarbeitung ablehnen.

O.Source_3 Fehlermeldungen und Log-Dateien DÜRFEN KEINE sensiblen Daten (z. B. User Identifier) enthalten.

O.Source_4 Potenzielle Ausnahmen im Programmablauf (Exceptions) MÜSSEN abgefangen, kontrolliert behandelt und dokumentiert werden. Technische Fehlerbeschreibungen (z.B. Strack Traces) DÜRFEN dem Nutzer NICHT angezeigt werden.

O.Source_5 Bei Ausnahmen im Programmablauf (Exceptions), mit sicherheitskritischen Auswirkungen, SOLL die Anwendung Zugriffe auf sensible Daten abbrechen und diese im Speicher sicher löschen.

O.Source_6 Bei Programmumgebungen mit manueller Speicherverwaltung (d.h., die Anwendung kann selbst exakt festlegen, wann und wo Speicher gelesen und beschrieben wird) MUSS die Anwendung für lesende und schreibende Zugriffe auf Speichersegmente auf sichere Funktionsalternativen (z. B. printf_s statt printf) zurückgreifen.

O.Source_7 Die Anwendung MUSS sicherstellen, dass alle sensiblen Daten unverzüglich nach der Erfüllung ihres Verarbeitungszwecks sicher gelöscht werden.

O.Source_8 Alle Optionen zur Unterstützung der Entwicklung (z. B. Log-Aufrufe, Entwickler-URLs, Testmethoden etc.) MÜSSEN in der Produktiv-Version mindestens deaktiviert oder besser vollständig entfernt sein.

O.Source_9 Der Hersteller MUSS sicherstellen, dass keinerlei Überreste von Debug-Mechanismen in der Produktiv-Version verbleiben.

O.Source_10 Die Implementierung der Anwendung SOLL moderne Sicherheitsmechanismen der Entwicklungsumgebung, wie beispielsweise Byte-Code-Minimierung und Stack-Protection, aktivieren.

A.4 Prüfaspekt (4): Drittanbieter-Software

O.TrdP_1 Der Hersteller MUSS eine zentrale und vollständige Liste von Abhängigkeiten durch externe Software, Bibliotheken und Frameworks führen.

O.TrdP_2 Externe Software, Bibliotheken und Frameworks MÜSSEN in der neusten oder der ihr vorhergehenden, verfügbaren „Stable“ -Version verwendet werden.

O.TrdP_3 Externe Bibliotheken und Frameworks MÜSSEN durch den Hersteller regelmäßig auf Schwachstellen überprüft werden. Der Hersteller MUSS ein Sicherheitskonzept vorlegen, das bei Bekanntwerden von Schwachstellen anhand der Kritikalität eine geduldete Weiternutzung für die Anwendung festlegt. Nachdem die Übergangsfrist (Grace Period) abgelaufen ist, DÜRFEN die betroffenen Funktionen aus Bibliotheken und Frameworks bei bekannten Schwachstellen NICHT eingesetzt werden.

O.TrdP_4 Sicherheitsupdates für externe Bibliotheken und Frameworks MÜSSEN zeitnah zur Verfügung gestellt werden. Der Hersteller MUSS ein Sicherheitskonzept vorlegen, das anhand der Kritikalität ausnutzbarer Schwachstellen die geduldete Weiternutzung für die Anwendung, bzw. das Hintergrundsystem festlegt. Nachdem die Übergangsfrist (Grace Period) abgelaufen ist, MUSS die Anwendung den Betrieb verweigern.

O.TrdP_5 Vor der Verwendung von externen Bibliotheken und Frameworks MUSS deren Quelle auf Vertrauenswürdigkeit geprüft werden.

O.TrdP_6 Die Anwendung SOLL sensible Daten nicht an Drittanbieter-Software weitergeben.

O.TrdP_7 Über Drittanbieter-Software eingehende Daten SOLLEN validiert werden.

O.TrdP_8 Drittanbieter-Software, die nicht länger vom Hersteller oder Entwickler gewartet wird, DARF NICHT verwendet werden.

A.5 Prüfaspekt (5): Kryptographische Umsetzung

O.Cryp_1 Beim Einsatz von Verschlüsselung in der Anwendung DÜRFEN KEINE fest einprogrammierten geheimen, bzw. privaten Schlüssel eingesetzt werden.

O.Cryp_2 Die Anwendung MUSS auf bewährte Implementierungen zur Umsetzung kryptographischer Primitive und Protokolle zurückgreifen (vgl. [TR02102-2]).

O.Cryp_3 Die Wahl kryptographischer Primitive MUSS passend zum Anwendungsfall sein und dem aktuellen Stand der Technik (siehe [TR02102-1]) entsprechen.

O.Cryp_4 Kryptographische Schlüssel DÜRFEN NICHT für mehr als genau einen Zweck eingesetzt werden.

O.Cryp_5 Die Stärke der kryptographischen Schlüssel MUSS dem aktuellen Stand der Technik entsprechen (siehe [TR02102-1]).

O.Cryp_6 Alle kryptographischen Schlüssel SOLLEN in einer vor Manipulation und Offenlegung geschützten Umgebung liegen.

O.Cryp_7 Alle kryptographischen Operationen SOLLEN in einer vor Manipulation und Offenlegung geschützten Umgebung stattfinden.

O.Rand_1 Alle Zufallswerte MÜSSEN über einen starken kryptographischen Zufallszahlengenerator erzeugt werden.

O.Rand_2 Die Anwendung MUSS Zufallszahlen von einem Zufallszahlengenerator mit hoher Entropie beziehen.

O.Rand_3 Die Anwendung SOLL einen Hardware-Zufallszahlengenerator verwenden, welcher keine Vergabe von Startwerten erlaubt. Stellt die Plattform einen solchen Hardware-Zufallszahlengenerator nicht zur Verfügung, SOLL die Anwendung dem Zufallszahlengenerator einen Startwert (Seed) zuweisen, der sich aus mindestens drei voneinander unabhängigen Systemparametern zusammensetzt. Die Parameter SOLLEN von außerhalb der Anwendung nicht ermittelbar und eindeutig sein.

O.Rand_4 Die Anwendung SOLL bei Erstellung eines Startwerts (Seed) für den Zufallszahlengenerator einen geeigneten Zufall aus einer geeigneten externen Quelle beziehen.

A.6 Prüfaspekt (6): Authentifizierung

O.Auth_1 Der Hersteller MUSS ein Konzept zur Authentifizierung (Zwei-Faktor-basiert), Autorisierung (Rollenkonzept) und zum Beenden einer Anwendungssitzung dokumentieren.

O.Auth_2 Die Anwendung SOLL Authentifizierungsmechanismen und Autorisierungsfunktionen separat realisieren. Sind für die Anwendung verschiedene Rollen notwendig, MUSS eine Autorisierung bei jedem Datenzugriff separat realisiert werden.

O.Auth_3 Jeder Authentifizierungsvorgang des Nutzers MUSS in Form einer Zwei-Faktor- Authentifizierung umgesetzt werden.

O.Auth_4 Für die Bewertung eines Authentifizierungsvorgangs SOLLEN zusätzliche Informationen (z. B. das verwendete Endgerät, der verwendete WiFi-Zugangsknoten oder die Zeit des Zugriffs) mit einbezogen werden.

O.Auth_5 Dem Nutzer SOLL eine Möglichkeit gegeben werden, sich über ungewöhnliche Anmeldevorgänge informieren zu lassen.

O.Auth_6 Die Anwendung MUSS Maßnahmen vorsehen, die ein Ausprobieren von Login-Parametern (z. B. Passwörter) verhindern.

O.Auth_7 Wurde die Anwendung unterbrochen (in den Hintergrundbetrieb versetzt), MUSS nach Ablauf einer angemessenen Frist (Grace Period) eine erneute Authentisierung durchgeführt werden.

O.Auth_8 Die Anwendung MUSS nach einer angemessenen Zeit in der sie nicht aktiv verwendet wurde (idle time) eine erneute Authentisierung fordern.

O.Auth_9 Die Anwendung MUSS nach einer angemessenen Zeit in der sie aktiv verwendet wurde (active time) eine erneute Authentisierung fordern.

O.Auth_10 Die Authentisierungsdaten DÜRFEN NICHT ohne eine ausreichende Authentifizierung des Nutzers geändert werden.

O.Auth_11 Die Anwendung MUSS für die Anbindung eines Hintergrundsystems eine geeignete Authentifizierung unterstützen.

O.Auth_12 Für die Nutzer-Authentifizierung in der Anwendungssitzung KANN der zweite Faktor vom Hintergrundsystem erzeugt werden.

O.Pass_1 Bei einer Authentifizierung mittels Benutzername und Passwort MÜSSEN starke Passwortrichtlinien existieren. Diese SOLLEN sich am aktuellen Stand gängiger „Best- Practices“ orientieren.

O.Pass_2 Für die Authentifizierung mittels Benutzername und Passwort KANN die Stärke des verwendeten Passworts dem Nutzer angezeigt werden. Informationen über die Stärke des gewählten Passworts DÜRFEN NICHT im Anwendungsspeicher oder im Hintergrundsystem persistiert werden.

O.Pass_3 Der Nutzer MUSS die Möglichkeit haben, sein Passwort zu ändern.

O.Pass_4 Das Ändern und Zurücksetzen von Passwörtern MUSS protokolliert werden. Der Nutzer SOLL über das Ändern und Zurücksetzten von Passwörtern informiert werden.

O.Pass_5 Werden Passwörter gespeichert, MÜSSEN diese mit einer den aktuellen Sicherheitsstandards entsprechenden Hash-Funktion und unter Verwendung geeigneter Salts gehasht werden.

O.Biom_1 Die Verwendung biometrischer Systeme DARF NICHT als alleiniger Authentifizierungsmechanismus eingesetzt werden. Sie ist lediglich als Teil einer Zwei-Faktor-Authentifizierung zulässig.

O.Biom_2 Der Hersteller MUSS definieren, welche Qualität und Eigenschaften ein biometrisches System mindestens aufweisen muss, um von der Anwendung verwendet werden zu dürfen. Diese Definition DARF NICHT unterhalb der in Anhang C beschriebenen Anforderungen des BSI liegen.

O.Biom_3 Vor jeder Authentifizierung über ein biometrisches System MUSS die Anwendung sicherstellen, dass die zur Verfügung stehende Hardware den festgelegten Anforderungen genügt.

O.Biom_4 Der Hersteller KANN biometrische Systeme verwenden, welche die Anforderungen aus O.Biom_2 nicht nativ erfüllen. In diesem Fall MUSS der Hersteller dafür Sorge tragen, dass die Anforderungen aus O.Biom_2 softwareseitig erfüllt werden.

O.Biom_5 Bevor die Anwendung ein biometrisches System nutzt, MUSS sie sicherstellen, dass dem System mindestens eine biometrische Referenz des Gerätenutzers zum Vergleich bereitsteht.

O.Biom_6 Die Anwendung MUSS feststellen, wenn die biometrische Referenz ohne Autorisierung verändert wurde und die Anmeldung ablehnen, falls biometrische Referenzen nachträglich (das heißt seit der Aktivierung des Authentifizierungskontrollmechanismus in der Anwendung) verändert worden sind.

O.Biom_7 Die Anwendung MUSS zur Auswertung der biometrischen Authentifizierung auf betriebssystemeigene Funktionalitäten zurückgreifen (z. B. Entsperren KeyChain/KeyStore).

O.Biom_8 Das biometrische System DARF NICHT mehr als fünf aufeinanderfolgende. fehlgeschlagene biometrischen Authentifizierungsversuche zulassen. Sobald dieser Schwellwert erreicht ist, MUSS die Authentifizierung mittels Biometrie deaktiviert werden. Eine erneute Nutzung ist erst nach erfolgreicher Authentifizierung mittels eines alternativen Authentisierungsfaktors möglich.

O.Sess_1 Das Session-Handling SOLL mittels sicherer Frameworks (vgl. O.Ntwk_3) realisiert werden.

O.Sess_2 Die Erstellung von Session-Identifiern MUSS durch den Zufallszahlengenerator des Hintergrundsystems erfolgen.

O.Sess_3 Session-Identifier MÜSSEN als sensible Daten geschützt werden.

O.Sess_4 Session-Identifier DÜRFEN NICHT unverschlüsselt auf permanenten Speichermedien abgelegt werden.

O.Sess_5 Die Anwendung MUSS es dem Nutzer ermöglichen einen oder alle zuvor ausgestellten Session-Identifier ungültig zu machen.

O.Sess_6 Wird eine Anwendungssitzung beendet, MUSS die Anwendung den Session-Identifier sicher löschen und das Hintergrundsystem informieren. Dies gilt sowohl für das aktive Beenden durch den Benutzer (log-out), als auch für das automatische Beenden durch die Anwendung (vgl. O.Auth_8 und O.Auth_9).

O.Tokn_1 Das Authentifizierungstoken MUSS auf dem Endgerät in einem sicheren Speicherbereich liegen (z. B. KeyChain/KeyStore).

O.Tokn_2 Es DÜRFEN KEINE sensiblen Daten in ein Authentifizierungstoken eingebettet werden.

O.Tokn_3 Ein Authentifizierungstoken MUSS ausschließlich die von der Anwendung erwarteten Felder enthalten.

O.Tokn_4 Der private Schlüssel zum Signieren des Authentifizierungstokens DARF NICHT auf dem Gerät vorliegen.

O.Tokn_5 Die Anwendung MUSS es dem Nutzer ermöglichen ein oder alle zuvor ausgestellten Authentifizierungstoken ungültig zu machen.

O.Tokn_6 Wird eine Anwendungssitzung beendet, MUSS die Anwendung den Authentifizierungstoken sicher löschen und das Hintergrundsystem informieren. Dies gilt sowohl für das aktive Beenden durch den Benutzer (log-out), als auch für das automatische Beenden durch die Anwendung (vgl. O.Auth_8 und O.Auth_9).

A.7 Prüfaspekt (7): Datensicherheit

O.Data_1 Die Werkseinstellung der Anwendung MUSS die maximale Sicherheit bieten.

O.Data_2 Sensible Daten MÜSSEN verschlüsselt gespeichert werden. Das Schlüsselmaterial für diese Verschlüsselung DARF NICHT unverschlüsselt persistiert werden. Dies gilt sowohl für flüchtiges Ablegen (z. B. im Arbeitsspeicher), als auch für dauerhaftes Speichern (z. B. in einer Cloud-Umgebung). Eine hardwareunterstützte Schlüsselverwaltung der Plattform SOLL bevorzugt verwendet werden.

O.Data_3 Die Anwendung SOLL sensible Daten in einer vor Einsicht und Manipulation geschützten Umgebung ablegen (z. B. embedded Secure Element/Trusted Execution Environment).

O.Data_4 Die Anwendung DARF Ressourcen, die einen Zugriff auf sensible Daten ermöglichen, gegenüber Dritten NICHT verfügbar machen.

O.Data_5 Alle erhobenen sensiblen Daten DÜRFEN NICHT über die Dauer ihrer jeweiligen Verwendung hinaus in der Anwendung gehalten werden.

O.Data_6 Die Anwendung MUSS die Grundsätze der Datensparsamkeit und Zweckbindung berücksichtigen.

O.Data_7 Die Speicherung und Verarbeitung von sensiblen Daten SOLL im Hintergrundsystem erfolgen.

O.Data_8 Bei der Verwendung von Aufnahmegeräten (z. B. Kamera) MÜSSEN sämtliche Metadaten mit Datenschutz-Relevanz, wie etwa Rückschlüsse auf die GPS-Koordinaten des Aufnahmeorts, eingesetzte Hardware etc., entfernt werden.

O.Data_9 Bei der Erhebung von sensiblen Daten durch die Verwendung von Aufnahmegeräten (z. B. Kamera), MUSS vorgebeugt werden, dass andere Anwendungen darauf Zugriff erlangen könnten, etwa über eine Mediengalerie.

O.Data_10 Bei der Eingabe sensibler Daten über die Tastatur SOLL die Anwendung unterbinden, dass Aufzeichnungen für Dritte erkennbar werden.

O.Data_11 Bei der Eingabe sensibler Daten SOLL der Export in die Zwischenablage unterbunden werden. Die Anwendung KANN alternativ eine eigene Zwischenablage implementieren, welche vor dem Zugriff durch andere Anwendungen geschützt ist.

O.Data_12 Sensible Daten wie biometrische Daten oder private Schlüssel DÜRFEN NICHT aus der Komponente, auf der sie erzeugt wurden, exportiert werden.

O.Data_13 Die Anwendung SOLL beim Anzeigen sensibler Daten den Zugriff für Dritte und die Speicherung des Bildschirms (z. B. Screenshots und Anzeigen für das App-Switching) unterbinden.

O.Data_14 Die Anwendung MUSS sicherstellen, dass im gesperrten Zustand des Endgeräts alle sensiblen Daten verschlüsselt sind.

O.Data_15 Die Anwendung MUSS lokal gespeicherte Daten mit einer sicheren Gerätebindung versehen.

O.Data_16 Schützt die Plattform das gewählte Speichermedium nicht vor Diebstahl (z.B. unverschlüsselte SD-Karten), MUSS die Anwendung bei einer Auswahl des betreffenden Speichermediums den Nutzer auf das erhöhte Risiko hinweisen.

O.Data_17 Die Anwendung MUSS sicherstellen, dass bei ihrer Deinstallation alle sensiblen Daten und anwendungsspezifischen Anmeldeinformationen auf dem Endgerät nicht mehr zugreifbar sind.

O.Data_18 Die Anwendung MUSS dem Nutzer die Möglichkeit geben, dass bei ihrer Deinstallation alle sensiblen Daten und anwendungsspezifischen Anmeldeinformationen vollständig gelöscht bzw. unzugänglich gemacht werden.

O.Data_19 Um dem Missbrauch von sensiblen Daten nach einem Geräteverlust entgegenzuwirken, KANN die Anwendung einen Kill-Switch realisieren, d. h. ein absichtliches, sicheres Überschreiben von Nutzerdaten im Gerät auf Anwendungsebene, ausgelöst durch das Hintergrundsystem. Der Hersteller MUSS die Auslösung des Kill-Switches durch den Anwender über das Hintergrundsystem durch starke Authentifizierungsmechanismen vor missbräuchlicher Nutzung schützen.

A.8 Prüfaspekt (8): Kostenpflichtige Ressourcen

O.Paid_1 Die Anwendung MUSS für den Nutzer kenntlich machen, welche kostenpflichtigen Leistungen (z.B. Zusatzfunktionalitäten oder Premiumzugriffe) und welche kostenpflichtigen Ressourcen (z.B. SMS, Telefonate, mobile Daten) von der Anwendung angeboten oder verwendet werden.

O.Paid_2 Die Anwendung MUSS vor der Verwendung kostenpflichtiger Leistungen das Einverständnis des Nutzers einholen.

O.Paid_3 Die Anwendung MUSS vor einer Zugriffsanforderung (z. B. Android-Berechtigungen) auf kostenpflichtige Ressourcen, das Einverständnis des Nutzers einholen.

O.Paid_4 Die Anwendung KANN für den Zugriff auf häufig verwendete, kostenpflichtige Ressourcen oder kostenpflichtige Leistungen ein dauerhaftes Einverständnis des Nutzers einholen, soweit dies dem primären Zweck der Anwendung angemessen ist.

O.Paid_5 Die Anwendung MUSS den Nutzer in die Lage versetzen zuvor erteilte Einverständnisse zurückzuziehen.

O.Paid_6 Die Anwendung SOLL die Transaktionshistorie von kostenpflichtigen Leistungen im Hintergrundsystem ablegen. Die Transaktionshistorie, einschließlich der Metadaten, MUSS als sensibles Datum gemäß O.Purp_8 behandelt werden.

O.Paid_7 Falls die Anwendung kostenpflichtige Leistungen anbietet, MUSS der Hersteller ein Konzept vorlegen, welches vorbeugt, dass Dritte die Zahlungsströme zur Nutzung von Anwendungsfunktionen zurückverfolgen können.

O.Paid_8 Die Anwendung MUSS dem Nutzer eine Übersicht der entstandenen Kosten anbieten. Falls die Kosten aufgrund einzelner Zugriffe erfolgt sind, MUSS die Anwendung einen Überblick der Zugriffe aufführen.

O.Paid_9 Die Validierung von getätigten Bezahlvorgängen SOLL im Hintergrundsystem vorgenommen werden.

O.Paid_10 Zahlverfahren von Drittanbietern MÜSSEN die Anforderungen an Drittanbieter-Software erfüllen.

A.9 Prüfaspekt (9): Netzwerkkommunikation

O.Ntwk_1 Jegliche Netzwerkkommunikation der Anwendung MUSS durchgängig mit TLS verschlüsselt werden.

O.Ntwk_2 Die Konfiguration der TLS-Verbindungen MUSS dem aktuellen Stand der Technik entsprechen und aktuellen Best-Practice-Empfehlungen folgen (vgl. [TR02102-2]).

O.Ntwk_3 Die Anwendung MUSS entweder die Sicherheitsfunktionalität der jeweilig verwendeten Betriebssystem-Plattform oder sicherheitsüberprüfte Frameworks oder Bibliotheken verwenden, um sichere Kommunikationskanäle aufzubauen.

O.Ntwk_4 Die Anwendung MUSS Zertifikats-Pinning unterstützen. Sie DARF Zertifikate NICHT akzeptieren, deren Zertifikatskette dem Hersteller nicht vertrauenswürdig erscheint [RFC7469].

O.Ntwk_5 Die Anwendung MUSS das Server-Zertifikat des Hintergrundsystems überprüfen.

O.Ntwk_6 Die Anwendung MUSS die Integrität und Authentizität der Antworten des Hintergrundsystems validieren.

O.Ntwk_7 Die Anwendung MUSS plattformspezifische Fallback-Mechanismen (z. B. „Cleartext Traffic Opt-out“ bzw. analog „In-App Transport Security“) deaktivieren.

O.Ntwk_8 Die Anwendung MUSS für alle aufgebauten Verbindungen Log-Dateien auf dem Hintergrundsystem vorhalten.

O.Ntwk_9 Ein abgebrochener Start MUSS als Sicherheitsereignis protokolliert werden. Die Protokollierung SOLL im Hintergrundsystem stattfinden.

A.10 Prüfaspekt (10): Plattformspezifische Interaktionen

O.Plat_1 Für die Nutzung der Anwendung SOLL das Endgerät über einen aktivierten Geräteschutz (Passwort, Mustersperre, o. ä.) verfügen. Im Fall eines nicht aktivierten Geräteschutzes MUSS der Hersteller den Nutzer über die damit verbundenen Risiken aufklären.

O.Plat_2 Die Anwendung DARF Berechtigungen, die für die Erfüllung ihres primären Zwecks nicht notwendig sind, NICHT einfordern.

O.Plat_3 Die Anwendung MUSS den Nutzer auf den Zweck der anzufragenden Berechtigungen und auf die Auswirkungen hinweisen, die eintreten, falls der Nutzer diese nicht gewährt.

O.Plat_4 Die Anwendung DARF KEINE sensiblen Daten in Meldungen oder Benachrichtigungen, die nicht vom Benutzer explizit eingeschaltet wurden (siehe O.Plat_5), schreiben.

O.Plat_5 Die Anwendung KANN dem Nutzer die Optionen bieten, Meldungen und Benachrichtigungen, ggf. auch mit sensiblen Daten, anzuzeigen. Bei Werkseinstellung MUSS diese deaktiviert sein.

O.Plat_6 Die Anwendung SOLL den Zugriff auf vorgesehene Dateipfade beschränken.

O.Plat_7 Die Anwendung MUSS Zugriffsbeschränkungen auf sämtliche Daten realisieren.

O.Plat_8 Die Anwendung MUSS Broadcast-Nachrichten auf autorisierte Anwendungen beschränken.

O.Plat_9 Die Anwendung DARF in Broadcast-Nachrichten KEINE sensiblen Daten versenden.

O.Plat_10 Das Anbieten von sensiblen Funktionalitäten über Interprozesskommunikation SOLL unterbunden werden. Ist das Anbieten zur Erfüllung des primären Zwecks erforderlich KANN eine solche Funktionalität angeboten werden. Alle angebotenen sensiblen Funktionalitäten MÜSSEN angemessen geschützt werden.

O.Plat_11 Die Anwendung SOLL verhindern, dass JavaScript während der Nutzung von WebView aktiv ist. Falls JavaScript für die Realisierung der Anwendung unabdingbar ist, MUSS die Anwendung JavaScript aus Quellen außerhalb der Kontrolle des Herstellers ablehnen.

O.Plat_12 Wechselt die Anwendung in den Hintergrundbetrieb, MUSS diese alle sensiblen Daten aus der aktuellen Ansicht („Views“ in iOS bzw. „Activities“ in Android) entfernen.

O.Plat_13 Die Anwendung MUSS alle nicht benötigten Protokoll-Handler in Web-Views deaktivieren.

O.Plat_14 Die Anwendung MUSS nach Beenden der Anwendung anwendungsspezifische Cookies gelöscht haben.

O.Plat_15 Die Anwendung SOLL nach Beenden alle nutzerspezifischen Daten im Arbeitsspeicher sicher überschrieben haben.

O.Plat_16 Der Nutzer MUSS über Sicherheitsmaßnahmen informiert werden, sofern diese durch den Nutzer umsetzbar sind.

A.11 Prüfaspekt (11): Resilienz

O.Resi_1 Die Anwendung MUSS dem Nutzer barrierearme Best-Practice-Empfehlungen zum sicheren Umgang mit der Anwendung und ihrer Konfiguration bereitstellen.

O.Resi_2 Die Anwendung MUSS Geräte mit Jailbreak oder Root-Zugriff entsprechend dem aktuellen Stand der Technik erkennen und angemessen darauf reagieren. Die Anwendung MUSS dem Nutzer darstellen, welche Risiken für die Daten des Nutzers bei einer Fortsetzung der Anwendung bestehen (z. B., dass diese offengelegt werden könnten) oder die Fortsetzung unterbinden.

O.Resi_3 Die Anwendung MUSS den Start in einer Entwicklungs-/Debug-Umgebung sicher erkennen und unterbinden.

O.Resi_4 Die Anwendung MUSS ihren Start abbrechen, falls sie unter ungewöhnlichen Benutzerrechten gestartet wird (z. B. root oder nobody).

O.Resi_5 Die Anwendung SOLL die Integrität des Endgeräts überprüfen, bevor sensible Daten verarbeitet werden.

O.Resi_6 Die Anwendung MUSS vor dem Zugriff auf das Hintergrundsystem dessen Authentizität überprüfen (siehe auch O.Ntwk_4).

O.Resi_7 Die Anwendung SOLL Härtungsmaßnahmen, wie etwa eine Integritätsprüfung vor jeder Verarbeitung sensibler Daten innerhalb des Programmablaufs, realisieren.

O.Resi_8 Die Anwendung MUSS starke Maßnahmen gegen Reverse Engineering umsetzen.

O.Resi_9 Die Anwendung MUSS Zugriffskontrollmechanismen unter der Berücksichtigung von unterschiedlichen Plattformen und Plattformversionen implementieren, so dass ein missbräuchlicher Zugriff auf Ressourcen durch eine Änderung der Plattformversion ausgeschlossen ist und somit in jeder Ausführungsumgebung ein hinreichender Schutz erzielt wird.

O.Resi_10 Die Anwendung MUSS robust gegenüber Störungen sein.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original