

Westsächsische Hochschule Zwickau

University of Applied Sciences

HOCHSCHULE FÜR MOBILITÄT | UNIVERSITY FOR MOBILITY

ANFORDERUNGEN UND LÖSUNGSWEGE FÜR CYBERSICHERHEIT IN MEDIZINPRODUKTSOFTWARE

Fakultät Physikalische Technik/Informatik
der Westsächsischen Hochschule Zwickau

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science

vorgelegt von

Christian Spitzer

geboren am 17.06.1997 in Landsberg am Lech

am 12.06.2025

Erstprüfer: Prof. Dr.-Ing. Thomas Franke
Zweitprüfer: Prof. Dr.-Ing. Rainer Wasinger
Sperrvermerk: Nein

Abstract

Die Sicherheit von Software ist insbesondere im Bereich der Medizinprodukte eine Herausforderung. Sensible Patientendaten bieten eine Angriffsfläche für Cyberangriffe, wodurch die Privatsphäre der Patienten gefährdet und ihre Sicherheit direkt beeinträchtigt werden kann. Es ist daher unerlässlich, mithilfe der regulatorischen Anforderungen robuste Cybersicherheitsmaßnahmen zu identifizieren und umzusetzen, um diese sensiblen Informationen effektiv zu schützen. Ziel der vorliegenden Arbeit ist es demnach, die Cybersicherheit bestehender Software zu erhöhen. Hierzu wird die zentrale Forschungsfrage „Welche aktuellen Anforderungen gibt es im Bereich Cybersicherheit speziell an Standalone-Medizinproduktsoftware und wie können diese bei der Weiterentwicklung von Produkten berücksichtigt werden?“ untersucht. Zur Beantwortung dieser Frage wurde eine detaillierte Analyse der regulatorischen Anforderungen aus Gesetzen, Verordnungen, Richtlinien und Normen durchgeführt. Ergänzt wurde die Analyse durch die prototypische Implementierung von Sicherheitsmaßnahmen wie Verschlüsselung, Kapselung externer Bibliotheken und Sandbox-Techniken. Darüber hinaus wurde die theoretische Umsetzung erweiterter Authentifizierung untersucht und die umgesetzten Maßnahmen evaluiert. Die Evaluation zeigt, dass der Einsatz des Advanced Encryption Standards mit Blockverkettung sowie Sandboxing-Techniken Medizinproduktsoftware sinnvoll vor Cyberangriffen schützen kann. Die vorliegende Arbeit hebt somit die Notwendigkeit von Cybersicherheitsstrategien hervor und bietet wertvolle Einblicke in mögliche Lösungen zum Schutz von Standalone-Medizinproduktsoftware sowie sensiblen Patientendaten in einem sich ständig wandelnden Bedrohungsumfeld.

Abkürzungsverzeichnis

AD	Zentraler Verzeichnisdienst (engl. Active Directory)
AES	Fortschrittlicher Verschlüsselungsstandard (engl. Advanced Encryption Standard)
API	Programmierschnittstelle (engl. Application Programming Interface)
BfArM	Bundesinstitut für Arzneimittel und Medizinprodukte
BSI	Bundesamt für Sicherheit in der Informationstechnik
CBC	Blockverkettung (engl. Cipher Block Chaining)
CEN	Europäisches Komitee für Normung (engl. European Committee for Standardization)
CENELEC	Europäisches Komitee für elektrotechnische Normung (engl. European Committee for Electrotechnical Standardization)
CPU	Zentrale Recheneinheit (engl. Central Processing Unit)
CRA	Cyberresilienz-Verordnung (engl. Cyber Resilience Act)
CVE	Häufige Schwachstellen und Gefährdungen (engl. Common Vulnerabilities and Exposures)
DACL	Benutzerdefinierte Zugriffskontrollliste (engl. Discretionary Access Control)
DDoS	Verteilte Dienstverweigerung (engl. Distributed Denial of Service)
DES	Datenverschlüsselungsstandard (engl. Data Encryption Standard)
DIN	Deutsches Institut für Normung
DiGAV	Verordnung über digitale Gesundheitsanwendungen
DLL	Dynamische Programmbibliothek (engl. Dynamic Link Library)
DSGVO	Datenschutz-Grundverordnung
DoS	Dienstverweigerung (engl. Denial of Service)

DVG	Digitale-Versorgungs-Gesetze
FDA	Lebensmittel- und Arzneimittelbehörde (engl. Food and Drug Administration)
FIDO2	Schnelle Online Identität 2 (engl. Fast IDentity Online 2)
FTA	Bundeshandelskommission (engl. Federal Trade Commission)
GG	Grundgesetz
HIPAA	Gesetz zur Übertragbarkeit und Rechenschaftspflicht von Krankenversicherungen (engl. Health Insurance Portability and Accountability Act)
IDE	Integrierte Entwicklungsumgebung
IEC	Internationale Elektrotechnische Kommission (engl. International Electrotechnical Commission)
IP	Internet Protokoll
IPC	Interprozesskommunikation
ISO	Internationale Organisation für Standardisierung (engl. International Organization for Standardization)
IV	Initialisierungsvektor
IVDR	Verordnung über Medizinprodukte für In-vitro-Diagnostika (engl. In vitro Diagnostic Medical Device Regulation)
MDCG	Koordinierungsgruppe Medizinprodukte (engl. Medical Device Coordination Group)
MDR	Medizinprodukte Verordnung (engl. Medical Device Regulation)
MEDDEV	Medizinische Geräteentwicklung (engl. Medical Device Development)
MPG	Medizin Produkt Gesetz
NIS	Netz- und Informationssicherheit (engl. Network and Information Security)

NIST	Nationales Institut für Normung und Technik (engl. National Institute of Standards and Technology)
PCB	Prozesskontrollblock (engl. Process Control Block)
PID	Prozesskennung (engl. Process identifier)
PIN	Persönliche Identifikationsnummer
SaMD	Software als Medizinprodukt (engl. Software as a Medical Device)
SBOM	Softwarestückliste (engl. Software Bill of Materials)
SID	Sicherheitskennung (engl. Security Identifier)
TCP	Übertragungssteuerungsprotokoll (engl. Transmission Control Protocol)
TPM	Vertrauenswürdige Plattformmodul (engl. Trusted-Platform-Modul)
UL	Unterzeichner-Laboratorien (engl. Underwriters Laboratories)
VPN	Virtuelles privates Netzwerk
V.T.O	Visuelles Behandlungsziel (engl. Visual Treatment Objective)

Inhaltsverzeichnis

1. Einleitung	1
2. Grundlagen	6
2.1. Begrifflichkeiten	6
2.1.1. Medizinprodukte	6
2.1.2. Gesetze, Verordnungen, Richtlinien und Normen	7
2.2. Software als Medizinprodukt	10
2.2.1. Aufgaben Medizinprodukt	10
2.2.2. Qualifizierung und Klassifizierung als Medizinprodukt	10
2.2.3. Regulatorische Anforderungen	13
2.3. Cybersicherheit	15
3. Methodik	20
3.1. Ermittlung von Cybersicherheitsanforderungen für OnyxCeph ^{3TM}	20
3.2. Ansätze für Schutzmaßnahmen	22
3.3. Auswahl von Schutzmaßnahmen	27
3.4. Planung der Auswertung	29
4. Umsetzung	30
4.1. Verschlüsselung	30
4.1.1. Windows Bitmap	31
4.1.2. Streaming-Klassen zum Speichern und Lesen von Bilddaten	32
4.1.3. Grundlagen der Verschlüsselung	33
4.1.4. Implementierung der Datenverschlüsselung	37
4.2. Kapselung externer Bibliotheken	45
4.2.1. Grundlagen von Prozessen unter Windows	45
4.2.2. Implementierung der Kapselung externer Bibliotheken	47
4.2.3. Implementierung von Sandboxing-Techniken	54

4.3. Authentifizierung	61
4.3.1. Grundlagen der Authentifizierung	61
4.3.2. Konzept zur Umsetzung einer Authentifizierungsmethode	62
5. Auswertung	69
6. Diskussion	73
7. Ausblick	76
A. Anhang	91
A.1. Auszug aus der grafischen Zusammenfassung des MDCG-Dokuments 2021-24 zur Qualifizierung/Klassifizierung von Software als Medizinprodukt . . .	91
A.2. Konkrete Anforderungen an Cybersicherheitsmaßnahmen nach Rechtsvorschrift für Software als Medizinprodukt	93
A.3. Auszug der Bilddaten einer Datei	94
A.4. Detaillierte Testergebnisse der Verschlüsselungsmaßnahmen	95
A.5. Detaillierte Testergebnisse der Kapselungsmaßnahmen	97
Danksagung	99

Abbildungsverzeichnis

1.1. Konzept der lokalen Netzwerkstruktur von OnyxCeph ^{3TM}	4
2.1. Pyramide des Rechts im europäischen Kontext	8
2.2. Ausprägungen von Software als Medizinprodukt	12
2.3. Entwicklung der Meldungen von Cybervorfällen der letzten zwölf Monate .	15
4.1. Hexadezimale Darstellung eines Pixelbildes	31
4.2. Formel zur Bestimmung der Arraygröße	32
4.3. Code zur Anpassung der Arraygröße auf ein Vielfaches der Blockgröße . . .	32
4.4. Datei mit Bilddaten unter Verwendung eines Streams erstellen	33
4.5. Symmetrische Verschlüsselung	35
4.6. Asymmetrische Verschlüsselung	36
4.7. Landschaftsbild im BMP-Format	37
4.8. Implementierung der XOR-Verschlüsselung	38
4.9. XOR-verschlüsseltes Landschaftsbild	39
4.10. AES-Verschlüsselung im CBC-Modus	41
4.11. Initialisierung der AES-Verschlüsselung	42
4.12. Blockweise AES-Verschlüsselung der Bilddaten	43
4.13. Abschluss der AES-Verschlüsselung	43
4.14. AES-256-CBC Verschlüsseltes Landschaftsbild	44
4.15. Zustandsautomat eines Prozesses	47
4.16. Vorbereitung des Argument-Strings für CreateProcessW	48
4.17. Aufruf der Funktion CreateProcessW	49
4.18. Erstellung einer anonymen Pipe	52
4.19. Starten des Kindprozesses mit CreateProcess	53
4.20. Sequenzielles Schreiben von Bilddaten in die Pipe	53
4.21. Erstellung eines eingeschränkten Tokens	55
4.22. Festlegung der Integritätsstufe auf Niedrig	56
4.23. Erstellung eines Auftragsobjekts mit DACL	57

4.24. Festlegung der Grenzen für das Auftragsobjekt	58
4.25. Ergebnisse der Sicherheits- und Funktionstests der Sandbox	60
4.26. Sequenzdiagramm zur FIDO2-Registrierung mit einem USB-Token	64
4.27. Sequenzdiagramm zur FIDO2-Authentifizierung mit einem USB-Token	66
5.1. Codebeispiel zur Zeitmessung mit QueryPerformanceCounter	70

Tabellenverzeichnis

3.1. Ermittlung von Cybersicherheitsmaßnahmen im Fall OnyxCeph ^{3TM}	21
5.1. Zusammengefasste Ergebnisse und Bewertungen der Sicherheitsmaßnahmen	72

1. Einleitung

In der allgegenwärtigen Digitalisierung von Daten und Prozessen und der immer stärkeren Vernetzung und Komplexität von Computersystemen kommt dem Thema Cybersicherheit eine stetig wachsende Bedeutung zu. Insbesondere Software in wichtiger Infrastruktur und mit Zugriff auf sensible Daten bietet ein vielversprechendes Ziel für verschiedenste Angreifer. Ein zentrales Beispiel dafür bildet das Gebiet der Medizinprodukte im Einsatz in Praxis- oder Kliniknetzwerken und in Kontakt mit Gesundheitsdaten der Patienten. Da es sich bei diesen Daten um persönliche Informationen handelt, muss ein effektiver Schutz vor potenziellen Angreifern bereitgestellt werden. Speziell für Medizinproduktsoftware ist es deshalb von großer Bedeutung, die aktuellen Anforderungen an die Cybersicherheit zu verstehen und umzusetzen.

Im Rahmen der Masterarbeit soll zunächst von den grundlegenden Begriffen der Cybersicherheit bis hin zu konkreten Normen und Richtlinien eine Übersicht der aktuellen Anforderungen speziell im Bereich der Standalone-Medizinproduktsoftware erarbeitet werden. Aufbauend darauf sollen für die vorgegebene Software zutreffende Regelungen und Ansätze identifiziert und strukturiert werden. Da es sich um ein etabliertes Softwareprodukt handelt, welches lediglich weiterentwickelt wird, sind für dieses Szenario geeignete Methoden für eine gezielte, softwareseitige Verbesserung der Cybersicherheit auszuwählen. Diese sollen prototypisch umgesetzt und in ihrer Wirkung untersucht werden. Ein wichtiges Ziel der Arbeit bildet die aus den gewonnenen Erkenntnissen abgeleitete Diskussion um theoretische Möglichkeiten und gleichzeitig praktische Grenzen der verschiedenen Ansätze der Cybersicherheit aufzuzeigen.

Software OnyxCeph^{3TM}

Die eigenständige (engl. standalone) Windows-Software OnyxCeph^{3TM} wurde von Image Instruments GmbH entwickelt und bietet Werkzeuge zur digitalen Umsetzung analoger, bildbasierter Arbeitsabläufe in kieferorthopädischen Praxen, Laboren und in universitären Einrichtungen. Die für diese Aufgaben entwickelten Werkzeuge sind flexibel konfigurierbar und kombinierbar, um eine Vielzahl von internen Arbeitsabläufen in der Kieferorthopädie digital abzubilden. Dadurch ergeben sich allgemeine Vorteile in Form von Zeit- und Kostenersparnis, Arbeitsvereinfachung und Effizienzsteigerung.

Ein wesentlicher Vorteil der Digitalisierung zeigt sich in der Abbildung von Gebissmodellen, die traditionell als physische Modelle aus Gips gefertigt wurden. Durch den Wegfall der Materialkosten für physische Modelle sinken die laufenden Kosten erheblich. Gleichzeitig verlagern sich die Arbeitsprozesse von der anspruchsvollen Laborarbeit hin zur Arbeit am Bildschirm, was den Arbeitsalltag erleichtert. Darüber hinaus ermöglicht die digitale Modellierung eine einfache Anpassung und Variation von Planungsmodellen, was zu mehr Flexibilität und Präzision in der Behandlungsplanung führt.

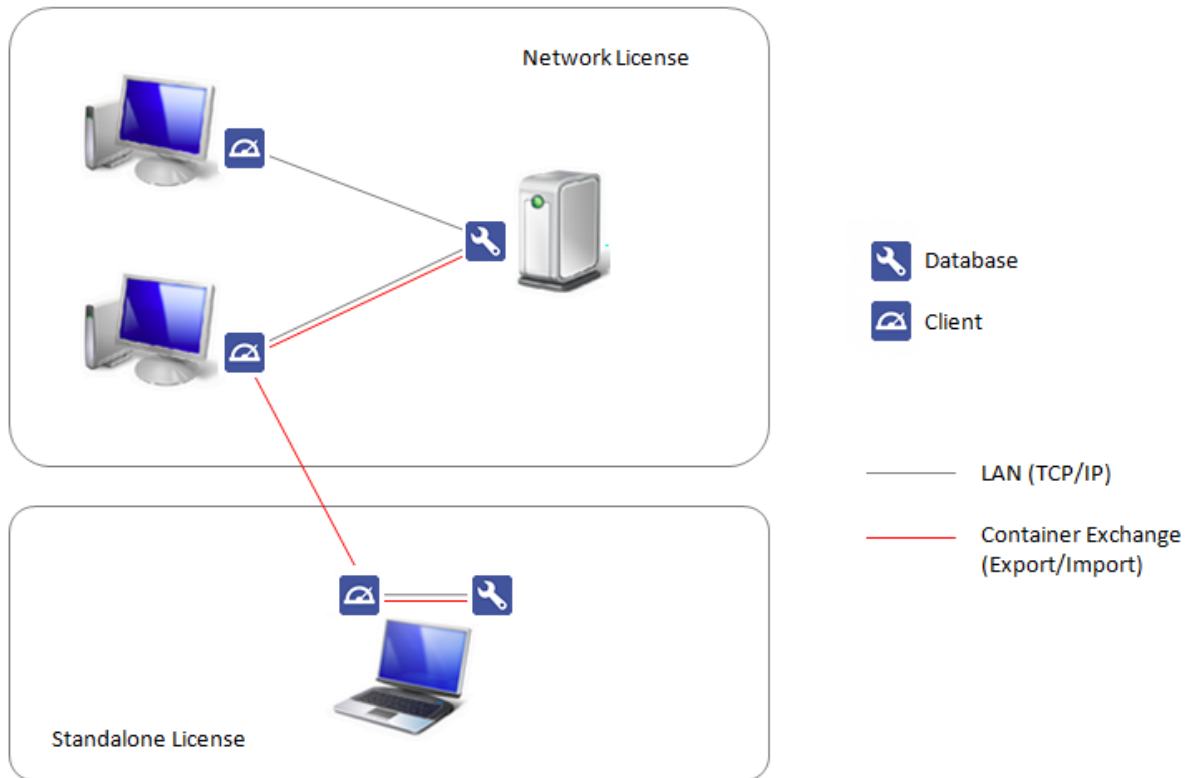
Neben den allgemeinen Vorteilen bietet die Software auch einen klinischen Nutzen durch die Ableitung metrischer Informationen aus fallrelevanten 2D- und 3D-Bilddaten, um beispielsweise die Länge und Breite eines Zahnes zu bestimmen. Aufgrund dieser Funktion wird OnyxCeph^{3TM} als Medizinprodukt klassifiziert. Die Software trifft jedoch keine Behandlungsentscheidungen und gibt keine Behandlungsempfehlungen. Ihre Ausgaben dürfen gemäß ihrer Zweckbestimmung nicht direkt für die Behandlung am Patienten verwendet werden. Die zur Verfügung gestellten Messwerte dienen lediglich dazu, bereits unabhängig getroffene Behandlungsentscheidungen entsprechend ihrer Zweckbestimmung abzusichern.

Die Anwendung unterstützt verschiedene bildbasierte kieferorthopädische Arbeitsabläufe, wie z.B. die metrische und anguläre Analyse von Bilddaten. Die Bilddaten können zur Archivierung, Vermessung, Planung und Beratung im Rahmen kieferorthopädischer Behandlungen verwendet werden. In OnyxCeph^{3TM} werden alle zweidimensionalen Bilder, die zu einem Patienten in der Datenbank vorhanden sind, nach Sitzungen (Aufnahmeserien) sortiert und als Vorschaubilder dargestellt. Daher ist Cybersicherheit notwendig, um sensible Patientendaten wie Name, Alter, Adresse und Bilder zu schützen. Alle regulären

Arbeitsabläufe werden in der Client-Software durchgeführt. Dafür ist diese modular gestaltet, wobei die folgende Liste eine Auswahl an Modulen zeigt, die für 2D- und 3D-Daten verfügbar sind: [Image Instruments, 2024b]

- Bild hinzufügen 2D: Bilder aus bildgebenden Systemen (wie von einer Digitalkamera) einlesen
- Kombinieren 2D: Bilder überlagern, bspw. Röntgenbild transparent mit Foto
- Auswertung 2D: Referenzpunkte am Bild setzen, um Strecken zu messen
- Bild hinzufügen 3D: Datensätze einlesen, die von 3D-Scannersystemen ausgegeben werden
- Modellausrichtung: Modelldatensätze digital beschneiden und mit Sockelschale versehen
- Segmentierung: Zerlegung des 3D-Scans in Einzelzähne
- V.T.O 3D: Aufstellung eines digitalen Behandlungsziels (Visual Treatment Objective)

OnyxCeph^{3TM} ist eine native Windows-Anwendung, die für den Einsatz in einem lokalen Client/Server-Netzwerk konzipiert ist. Technisch gesehen handelt es sich hierbei um eine netzwerkbasierte Windows-Anwendung, die aus einem zentralen Datenbankserver (Server) und einem oder mehreren verbundenen Arbeitsplätzen (Clients) besteht, wie in Abbildung 1.1 dargestellt.



Quelle: [Image Instruments, 2024a]

Abbildung 1.1.: Konzept der lokalen Netzwerkstruktur von OnyxCeph^{3TM}

Die Kommunikation zwischen Server und Clients erfolgt über das lokale Netzwerk mittels TCP/IP-Protokolls und eines benutzerspezifischen Ports. Obwohl eine Verbindung über das Internet technisch möglich wäre, wird diese Art der Anbindung von den Systemvoraussetzungen der Software nicht unterstützt. [Image Instruments, 2024d] Aus diesem Grund ist OnyxCeph^{3TM} als sogenannte „On-Premise“-Lösung konzipiert, das heißt als „Vor-Ort“-Lösung im Gegensatz zu einer Web- oder Cloudbasierten Lösung. Der datenbasierte Server speichert alle relevanten Informationen, von den eigentlichen 2D- oder 3D-Daten bis hin zu den individuellen Einstellungen der Arbeitsplätze (z.B. Fenstergröße, Anordnung oder Farbschema). Zu beachten ist, dass Verbindungen zum Server über VPN nicht unterstützt werden. Das System besteht also aus einem lokalen Netzwerk-Datenbankserver und einer oder mehreren Arbeitsstationen, die innerhalb dieses Netzwerks integriert sind. Die Software arbeitet ausschließlich innerhalb des lokalen Netzwerks und schließt damit die Mehrzahl der Schwachstellen und Angriffsszenarien der Online-Kommunikation aus, weshalb dieser Bereich für die weitere Aufgabenstellung nicht betrachtet wird. [Image Instruments, 2024c]

Auf Windows-Arbeitsplätzen kann die Software als normale Anwendung installiert und ohne besondere administrative Rechte betrieben werden. Auch ohne besondere Rechte bei der Ausführung der Software und trotz der strikten Trennung von Datenbank und Arbeitsplätzen stellt der Betrieb der Software eine zusätzliche Komponente in der IT-Infrastruktur dar und muss sich mit Themen von Datenschutz bis hin zur Sicherheit gegenüber Cyberangriffen auseinandersetzen. Die diesbezüglichen Anforderungen ergeben sich im Wesentlichen aus den gesetzlichen Vorgaben, die auf Anwenderseite einzuhalten sind, oder aus den konkreten Anforderungen, die sich aus der Einstufung als Medizinprodukt ergeben.

2. Grundlagen

Cyberangriffe, Medizinprodukteverordnung und regulatorische Anforderungen sind Beispiele für Begriffe, die für das Thema der Arbeit von großer Bedeutung sind. Daher werden im folgenden Kapitel die zentralen Fachbegriffe für die vorliegende Arbeit einheitlich eingeführt. Die Grundlagen dienen somit dem Verständnis und der Eingrenzung des umfangreichen Themas.

2.1. Begrifflichkeiten

In diesem Abschnitt werden die im weitergehenden Text verwendeten Begriffe Medizinprodukt sowie die rechtlichen Grundlagen beschrieben, abgegrenzt und mit ihren Abhängigkeiten dargestellt.

2.1.1. Medizinprodukte

Medizinprodukte spielen eine entscheidende Rolle im Gesundheitswesen und sind unverzichtbare Hilfsmittel für die Diagnose und Behandlung von Patienten. Sie sind spezialisierte Produkte, die für medizinische Zwecke am Menschen verwendet werden und hauptsächlich physikalisch wirken. Sie umfassen ein breites Spektrum von Anwendungen wie:

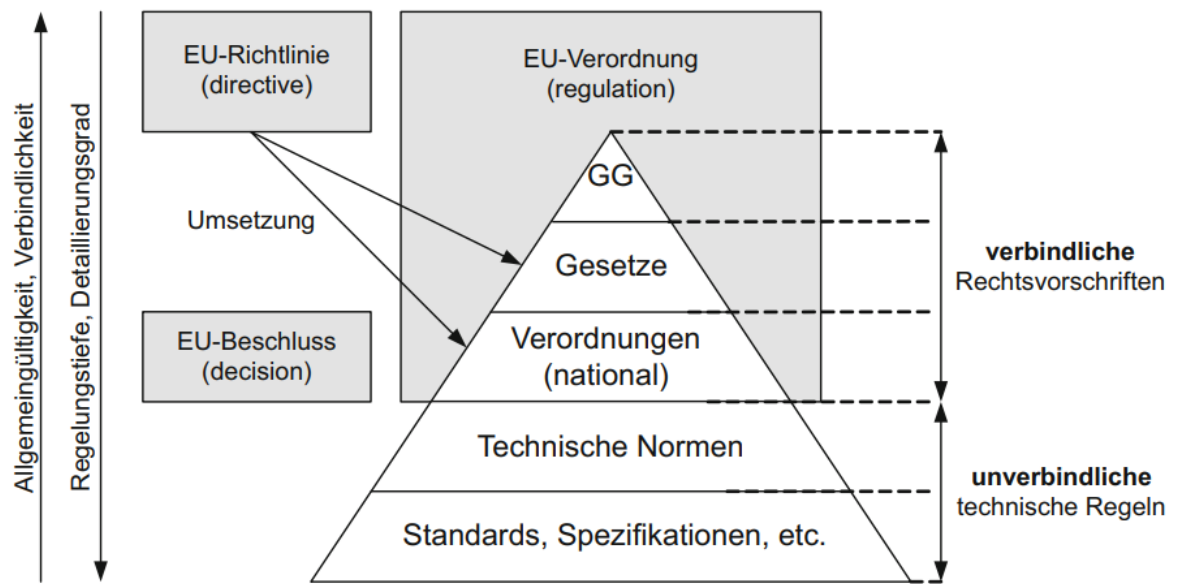
- Implantate
- Instrumente für Injektionen
- Infusionen
- Transfusionen und Dialyse
- Herzschrittmacher
- Sehhilfen
- Software

In diese Kategorie fallen auch Labordiagnostika und In-vitro-Diagnostika, wie Reagenzien, Kits und Geräte, die zur Untersuchung von Proben aus dem menschlichen Körper bestimmt sind. Darüber hinaus gibt es Medizinprodukte, die Substanzen enthalten oder mit Substanzen beschichtet werden, die bei gesonderter Verwendung als Arzneimittel gelten. Des Weiteren gelten Produkte zur Empfängnisverhütung oder -förderung und Produkte, welche speziell für die Reinigung, Desinfektion oder Sterilisation dienen, als Medizinprodukt.

Die ordnungsgemäße Aufbereitung zur sterilen und keimarmen Anwendung von Medizinprodukten ist in der Medizinprodukte-Betreiberverordnung geregelt. Insbesondere für Implantate bestehen Betreiberpflichten, welche die Stärkung der Patientensicherheit erhöhen. [BMG, 2025] Die Gewährleistung der Patientensicherheit ist ein wesentlicher Bestandteil der Cybersicherheit. Um eine einheitliche Umsetzung zu gewährleisten, werden gesetzliche Regelungen und Leitfäden herangezogen, die im folgenden Kapitel beschrieben werden.

2.1.2. Gesetze, Verordnungen, Richtlinien und Normen

Unternehmen, die Software als Medizinprodukt herstellen, stehen vor der Herausforderung, sich nicht nur gegen eine Vielzahl von Cyberbedrohungen zu schützen. Gleichzeitig muss sichergestellt werden, dass die Sicherheitsmaßnahmen den gesetzlichen Anforderungen entsprechen. Hierfür sind Gesetze, Verordnungen, Richtlinien und Normen relevant. Die Rechtsverordnungen stehen in Beziehung zueinander und bilden zusammen mit dem Grundgesetz (GG) die Rechtsordnung. Zur Veranschaulichung dient die Abbildung 2.1, die die Hierarchie der Rechtsformen in Form einer Pyramide darstellt. An der Spitze steht die Rechtsform mit dem höchsten Verbindlichkeits- und Detaillierungsgrad. In der Abbildung ist dies das Grundgesetz. An der Basis steht die Rechtsform mit dem geringsten Verbindlichkeits- und Detaillierungsgrad.



Quelle: [Schnieder and Schnieder, 2013, S. 593]

Abbildung 2.1.: Pyramide des Rechts im europäischen Kontext

Gesetze

Gesetze sind rechtlich verbindliche Regelungen, die das geordnete Zusammenleben einer Gesellschaft sicherstellen. In Deutschland unterscheidet man zwischen formellen und materiellen Gesetzen. Formelle Gesetze werden im verfassungsrechtlich vorgeschriebenen Gesetzgebungsverfahren erlassen. Sie werden auf Bundesebene vom Bundestag und auf Landesebene von den Landtagen verabschiedet. Die Einhaltung der Gesetze ist Pflicht. Ein Beispiel aus dem Gesundheitsbereich ist das Medizinproduktegesetz (MPG) in Deutschland. Dieses Gesetz bildet den rechtlichen Rahmen für die Sicherheit, Qualität und Leistung von Medizinprodukten. [BT, 2024]

Verordnungen

Gesetze werden von der Legislative, dem Parlament, erlassen. Verordnungen hingegen werden von der Exekutive, also der Verwaltung, erlassen. Dieser Unterschied ist von großer Bedeutung, da Gesetze die Rahmenbedingungen und grundsätzlichen Vorhaben festlegen, während Verordnungen konkretisieren, wie diese gesetzlichen Vorgaben in der Praxis umzusetzen sind. Eine Verordnung gibt also detaillierte Anweisungen, wie ein bereits bestehendes Gesetz vollständig auszuführen ist. In den Gesetzen selbst sind diese

Details oft nicht festgelegt. Würde das Parlament versuchen, alle Einzelheiten eines Gesetzes genau zu regeln, würde dies den Gesetzgebungsprozess erheblich verlängern und die rechtzeitige Verabschiedung neuer Gesetze behindern. Eine Verordnung ist somit ein verbindlicher Rechtsakt, der von allen beteiligten Ländern vollständig umgesetzt werden muss. [Christiane Toyka-Seid, 2024] Ein Beispiel ist die MDR, eine EU-Verordnung über Medizinprodukte. Diese Verordnung legt eine Vielzahl von Anforderungen fest, die Hersteller erfüllen müssen, um Medizinprodukte in der EU sicher und qualitativ hochwertig in Verkehr bringen zu dürfen. [EU, 2017]

Richtlinien

Eine Richtlinie ist ein Rechtsakt, der ein Ziel definiert, das von den Mitgliedstaaten erreicht werden soll. Es ist jedoch Sache der einzelnen Länder, eigene Gesetze zu erlassen, um dieses Ziel zu erreichen. [EU, 2024] Beispielsweise legt die Richtlinie über Netz- und Informationssicherheit (NIS) Maßnahmen zur Gewährleistung eines hohen gemeinsamen Sicherheitsniveaus von Netz- und Informationssystemen innerhalb der Europäischen Union fest. Sie schafft einen einheitlichen Rechtsrahmen für den Aufbau nationaler Cybersicherheitskapazitäten in der EU. [BSI, 2024b]

Normen

Eine Norm ist eine allgemein gültige Regel, die in einem festgelegten Normungsverfahren beschlossen und rechtlich anerkannt wird. Sie wird von Ausschüssen im Konsens erarbeitet, muss technisch ausgereift sein und dem Anwender einen Nutzen bringen. Normen sind von anerkannten Institutionen angenommene Dokumente, die für die allgemeine und wiederkehrende Anwendung Regeln, Richtlinien oder Merkmale für Tätigkeiten oder deren Ergebnisse festlegen. Dabei wird immer ein optimaler Ordnungsgrad in einem gegebenen Kontext angestrebt. Die Anwendung von Normen ist grundsätzlich freiwillig, was einen wesentlichen Unterschied zu Gesetzen und Richtlinien darstellt, die rechtlich bindend sind. [IHK Rhein-Neckar, 2024] Die International Organization for Standardization (ISO) und die International Electrotechnical Commission (IEC) sind die beiden wichtigsten internationalen Normungsorganisationen im europäischen Kontext, während das European Committee for Standardization (CEN) und das European Committee for Electrotechnical Standardization (CENELEC) für die Normung auf europäischer Ebene zuständig sind. [DIN, 2025][Hastenteufel et al., 2019] Ein konkretes Beispiel für eine Norm ist die ISO 27001. Sie legt die Grundsätze und Anforderungen fest, nach denen Unternehmen handeln

müssen, wenn sie ein IT-Sicherheitsmanagementkonzept für Medizinprodukte umsetzen und normkonform einführen wollen. [DIN, 2024]

2.2. Software als Medizinprodukt

Um eine Software als Medizinprodukt klassifizieren zu können, muss zunächst geklärt werden, welche Aufgaben ein Medizinprodukt im Allgemeinen erfüllt. Anschließend wird ermittelt, welche zusätzlichen Anforderungen an eine Software als Medizinprodukt zu stellen sind und unter welchen Bedingungen diese erfüllt sind.

2.2.1. Aufgaben Medizinprodukt

Medizinprodukte haben eine Vielzahl an Aufgaben, um spezifische Zwecke zu erfüllen. Zur Definition der Aufgaben wird die Medical Device Regulation (MDR) herangezogen. Darin ist festgelegt, woran sich Hersteller zu halten haben. Insbesondere Artikel 2 der MDR fasst die relevanten Aufgaben wie folgt zusammen: Medizinprodukte sind essenziell für die Erkennung und Überwachung von Krankheiten sowie deren Vorhersage und Prognose von Gesundheitszuständen. Zudem werden sie häufig zur direkten Behandlung oder Linderung von Krankheiten, Verletzungen oder Behinderungen eingesetzt. Neben der Behandlung und Linderung dienen Medizinprodukte auch der Kompensation von Behinderungen. Medizinprodukte werden auch eingesetzt, um die menschliche Anatomie oder physiologische und pathologische Prozesse zu untersuchen, zu ersetzen oder zu verändern. Eine weitere Aufgabe ist die Gewinnung von Informationen durch die In-vitro-Untersuchung von Proben, die aus dem menschlichen Körper stammen, wie Blut, Gewebe oder Organspenden. Diese Untersuchung ist entscheidend für die genaue Bestimmung von Krankheiten. [Hastenteufel et al., 2019] [BfArM, 2024]

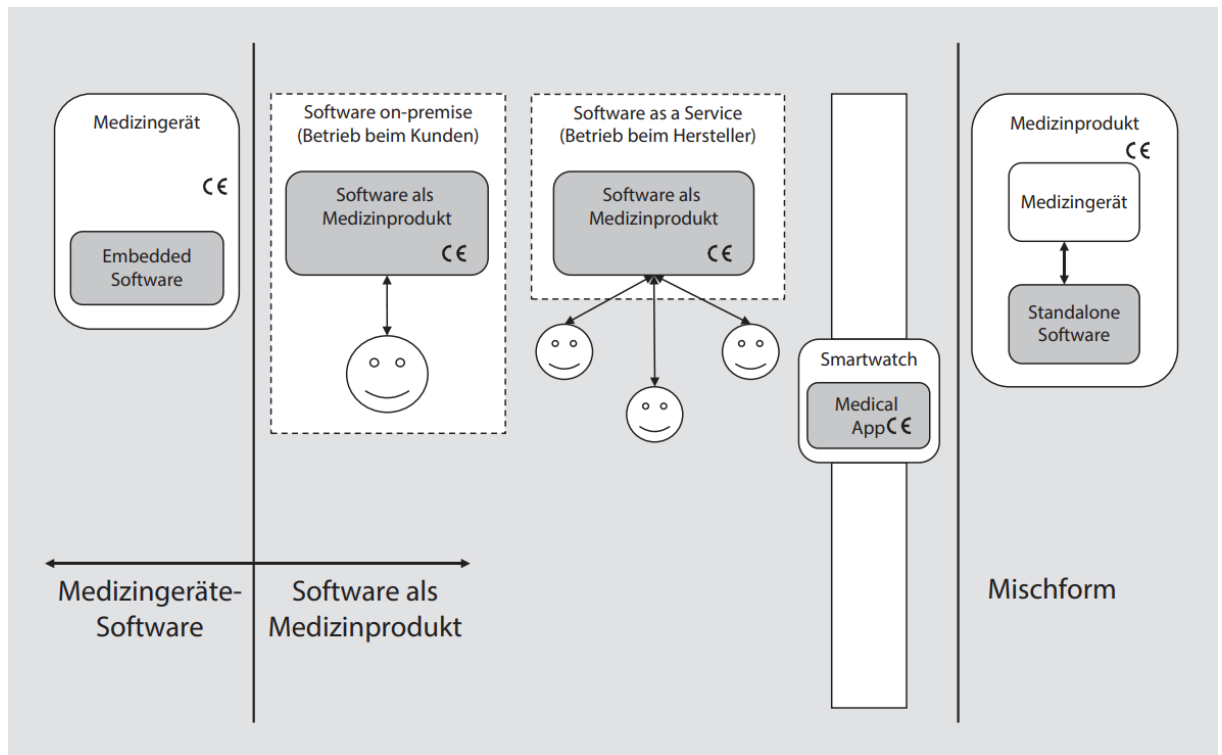
2.2.2. Qualifizierung und Klassifizierung als Medizinprodukt

Weder das Medizinproduktegesetz noch die Medizinprodukte-Betreiberverordnung präzisieren und differenzieren das Medizinprodukt Software. Es liegt somit im Ermessen des Herstellers, ob Software über die Zweckbestimmung als Medizinprodukt definiert wird oder nicht. [Gärtner, 2010] Um jedoch den Begriff Medizinprodukt Software definieren zu können, gibt das International Device Regulators Forum folgende Definition vor: „Software als Medizinprodukt (SaMD) ist definiert als Software, die für einen oder mehrere

medizinische Zwecke bestimmt ist und diese Zwecke erfüllt, ohne Teil einer medizinischen Hardware zu sein.“ Zusätzliche Anmerkungen präzisieren die Definition wie folgt: [IMDRF, 2013]

- SaMD ist ein Medizinprodukt und schließt ein In-vitro-Diagnostikum ein.
- SaMD können auf allgemeinen (nicht medizinischen) Computerplattformen betrieben werden.
- Software entspricht nicht der Definition eines SaMD, wenn ihr bestimmungsgemäßer Zweck darin besteht, ein Hardware-Medizinprodukt zu steuern.
- SaMD können in Kombination (z.B. als Modul) mit anderen Produkten, einschließlich Medizinprodukten, verwendet werden.
- SaMD können mit anderen Medizinprodukten verbunden werden, einschließlich Hardware-Medizinprodukten und anderer SaMD-Software.
- Mobile Anwendungen, die der obigen Definition entsprechen, gelten als SaMD.

SaMD bezeichnet also Software, die unter die Definition eines Medizinprodukts fällt und nicht als eingebettete Software integraler Bestandteil eines Medizinprodukts ist. Software als Medizinprodukt, die auf Standardendgeräten wie Servern, Desktop-PCs, Smartphones und anderen läuft, wird allgemein als „Standalone-Software“ bezeichnet. [Hastenteufel et al., 2019] Es gibt aber auch Mischformen, die sowohl als eigenständige Software als auch als Medizinprodukt zugelassen sind. Ein Beispiel wäre eine Bedienkonsole für einen Computertomographen. Um die verschiedenen Kategorien besser einordnen zu können, gibt Abbildung 2.2 einen Überblick über die Arten und Ausprägungen von Softwarelösungen in Medizinprodukten.



Quelle: [Hastenteufel et al., 2019, S. 152]

Abbildung 2.2.: Ausprägungen von Software als Medizinprodukt

Software als Medizinprodukt hat somit eine Vielzahl von Aufgabenbereichen, die vom Bundesinstitut für Arzneimittel und Medizinprodukte (BfArM) wie folgt beschrieben werden: Alarmieren, Analysieren, Berechnen, Detektieren, Diagnostizieren, Interpretieren, Konvertieren, Messen, Steuern, Überwachen und Verstärken.

Ist die Entscheidung gefallen, dass eine Software als Medizinprodukt eingesetzt werden soll, muss sie als solches klassifiziert werden. Die MDR definiert hierzu 22 Klassifizierungsregeln, die für Software gelten und diese in vier Klassen einteilen. Die Klassen sind gemäß Artikel 51 MDR in I, IIa, IIb und III definiert, wobei I die niedrigste Sicherheitsstufe und III die höchste Sicherheitsstufe darstellt. [BfArM, 2024] Die Klassifizierungsregeln können dabei unterteilt werden in: [Hastenteufel et al., 2019]

- die Nutzungsdauer, in der die Nutzungszeiträume definiert sind.
- die Art der Invasivität, wobei zwischen dem Eindringen eines Produkts durch eine Körperöffnung oder durch einen chirurgischen Eingriff zu unterscheiden ist.
- die Energieversorgung, wobei zwischen Medizinprodukten mit externer Energiequel-

le und Medizinprodukten ohne externe Energiequelle unterschieden wird.

- der Körperteil, mit dem das Produkt in Berührung kommt.

Zur Unterstützung der Entscheidung über die Qualifizierung und Klassifizierung als Medizinprodukt können eine Reihe von Richtlinien herangezogen werden. Eine Entscheidungshilfe zur Qualifizierung und Klassifizierung von Software als Medizinprodukt ist die insbesondere in Europa relevante MDCG-Richtlinie 2021-24 [EU, 2025a], welche Software als Medizinprodukt definiert. Zusätzlich zu seinen Vorgängern enthält dieses Dokument ein mehrseitiges Diagramm (siehe Auszug im Anhang A.1), das bei der Entscheidung über die Einstufung einer Software als Medizinprodukt und deren Einordnung in die entsprechenden Klassen I, IIa, IIb und III hilft. Weitere Entscheidungshilfen für die Qualifizierung und Klassifizierung von Software als Medizinprodukt sind unter anderem: [Wassel, 2025]

- EU: Manual on Borderline and Classification
- International Medical Device Regulator Forum
- Britische Medicines and Healthcare products Regulatory Agency (MHRA)
- Food and Drug Administration (FDA)

Erfüllt eine Software nun die notwendigen Kriterien und wird als Medizinprodukt eingestuft, sind weitere regulatorische Anforderungen notwendig, um die Software auch nach außen hin als adäquat zu repräsentieren.

2.2.3. Regulatorische Anforderungen

Regulatorische Anforderungen werden genutzt, um von den zuständigen Behörden und den Kunden als seriöses Unternehmen wahrgenommen zu werden. Dabei wird auf bestehende Gesetze, Verordnungen, Richtlinien und Normen zurückgegriffen. Die gängige Praxis für die Umsetzung wird im Folgenden für den Fall von OnyxCeph^{3TM} als „Standalone Software“ beschrieben. Aufgrund der Einstufung als SaMD müssen zunächst das Medizinproduktegesetz und die Medizinprodukte-Betreiberverordnung umgesetzt werden. Die In vitro Diagnostic Medical Device Regulation (IVDR) ist in bestimmten Fällen ebenfalls anzuwenden. Verbindlich sind auch die europäischen Vorgaben der Datenschutz-Grundverordnung (DSGVO), das Digitale-Versorgungs-Gesetz (DVG) und der Verordnung über digitale Gesundheitsanwendungen (DiGAV). Für die USA wären dies wiederum die Cybersecurity Guidance Documents der FDA, der HIPAA Act, Health Breach

Notification Rule und die Vorschriften der Federal Trade Commission (FTC). Die MDR verweist auch auf die Anwendung von Normen zur Sicherstellung der Konformität. Die wichtigsten Normen laut Hastenteufel und Renaud (2019) für eine SaMD sind:

[Hastenteufel et al., 2019]

- ISO 14971 - Risikomanagement [DIN, 2022]
- ISO 13485 - Qualitätsmanagement [DIN, 2021]
- IEC 62304 - Software-Lebenszyklus [DIN, 2003]
- IEC 62366-1 - Usability [DIN, 2015]

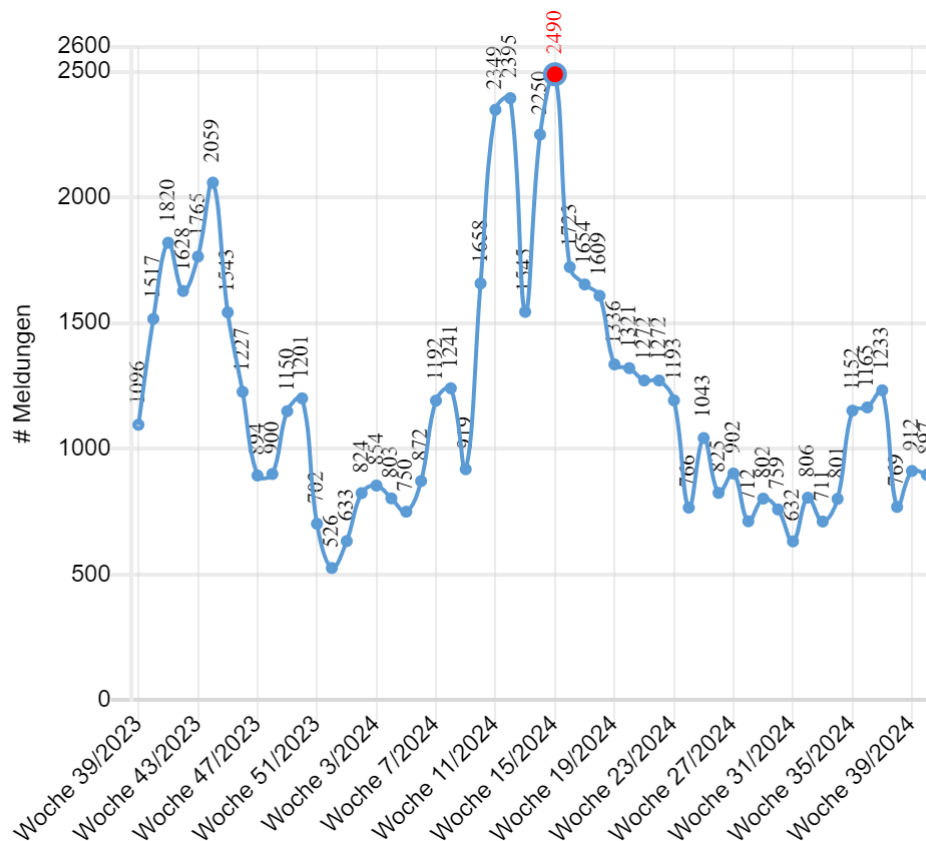
Das Johner Institut (2024) erweitert die Liste der relevanten Normen zur Sicherstellung des Standes der Technik um folgende Normen: [Johner Institut, 2025]

- IEC 81001-5-1 - Die harmonisierte Norm für sichere Health Software
- ISO 29147 - Wie Hersteller IT-Schwachstellen offenlegen sollten
- IEC 60601-4-5 - Die Norm zur IT-Sicherheit auch für Standalone-Software
- ISO 27001: IT-Sicherheitsmanagement für alle Medizinproduktehersteller
- IEC 62443-4-1: IT-Sicherheit als Teil des Produktlebenszyklus
- UL 2900 – Weshalb Sie den IT-Security-Standard kennen, aber niemals kaufen sollten
- ISO/IEC 15408: IT-Sicherheit von (Medizin-)Produkten bewerten

Neben den Normen ist insbesondere die Richtlinie Medical Device Development (MED-DEV) zu nennen, die jedoch weitgehend veraltet ist und durch die Medical Device Coordination Group (MDCG) ersetzt wurde. [EU, 2025a] Dies sind frei zugängliche Leitfäden der Europäischen Kommission für Hersteller zur Umsetzung der MDR. Die aktuellste Erweiterung der Cybersicherheit ist der Cyber Resilience Act (CRA), der Ende 2024 in Kraft trat und bis 2027 umgesetzt werden muss. Der CRA sieht den Nachweis von Anforderungen, die Offenlegung von Schwachstellen und die Gewährleistung der Sicherheit während des Unterstützungszeitraums vor. [BSI, 2025b] Schließlich sind die Standards und Richtlinien des National Institute of Standards and Technology (NIST) zu nennen. Das NIST soll bei der Bewältigung von Informationssicherheitsrisiken helfen und baut auf bestehenden Standards und Richtlinien auf. Zu diesen Standards gehört z.B. NIST SP 800-53, eine Sammlung von möglichen Sicherheitsmaßnahmen. [Hastenteufel et al., 2019]

2.3. Cybersicherheit

Die Digitalisierung bringt viele Vorteile und neue Möglichkeiten mit sich. Damit steigt aber auch die Dringlichkeit für die Sicherheit der SaMDs. [Schünemann, 2023] Das Bundesamt für Sicherheit in der Informationstechnik hat daher ein Lagebild zur IT-Sicherheit für Deutschland im Jahr 2023 erstellt. Demnach wurden im Berichtszeitraum täglich rund 250.000 neue Schadprogrammvarianten identifiziert, 21.000 Systeme mit Schadsoftware infiziert und gemeldet, 775 E-Mails mit Schadprogrammen abgefangen und 370 Webseiten für den Zugriff aus Regierungsnetzen gesperrt. [BSI, 2024a] Auch das Bundesamt für Cybersicherheit BACS unseres Nachbarlandes Schweiz zeigt in Abbildung 2.3 die Anzahl der wöchentlich gemeldeten Cybervorfälle. Allein in der Woche 15/2024 wurden beispielsweise 1993 Betrugsversuche, 311 Phishing-Angriffe, 4 Schadsoftware-Angriffe und weitere Angriffe gemeldet. [BACS, 2024]



Quelle: [BACS, 2024]

Abbildung 2.3.: Entwicklung der Meldungen von Cybervorfällen der letzten zwölf Monate

Die aktuellen Beispiele zeigen, wie akut Cyberkriminalität ist. Deshalb ist es hilfreich, sich in die Lage eines Cyber-Angreifers zu versetzen. Denn Cyber-Angreifer verfolgen eine zentrale Frage: „Wie kann ich mit minimalem Aufwand und unerkannt an wichtige Assets gelangen?“ [Pufahl et al., 2024] Mit Assets sind nicht physische Gegenstände gemeint, sondern insbesondere digitale Daten, Managementsysteme, Unternehmensressourcen und Rechnerkapazitäten. Die Ziele eines Cyber-Angriffs können unterschiedlicher Natur sein und schließen einander nicht aus. [Pufahl et al., 2024]

- Datendiebstahl: Ein Cyber-Angreifer sucht und kopiert Daten wie Kunden-, Finanz-, Personaldaten oder Urheberrechte, um sie weiterzuverkaufen.
- Spionage: Industrielle oder geopolitische Spionage.
- Erpressung: Ziel ist die Verschlüsselung der Daten und möglichst auch des Backups (Ransomware-Angriff), um Geld für die Entschlüsselung zu erpressen.

Cyber-Angriffe mit diesen Zielen nehmen viele verschiedene Formen an. Jede Angriffsart hat ihre eigenen Techniken und Ziele. Nachfolgend sind einige der häufigsten Arten von Cyber-Angriffen laut dem Bundesamt für Sicherheit in der Informationstechnik (BSI) zusammengefasst: [BSI, 2025a]

- Denial of Service (DoS) und Distributed Denial of Service (DDoS):
Diese Angriffe zielen darauf ab, die Verfügbarkeit von Netzen oder Diensten durch Überlastung zu verhindern. Ein DoS-Angriff wird von einem Punkt aus initiiert, während ein DDoS-Angriff koordiniert von vielen Punkten aus durchgeführt wird, um Systeme lahm zu legen. [BSI, 2025e]
- Schadsoftware (Malware):
Diese Art von Software wird unerlaubt auf Systemen installiert. Ziel ist es, Daten zu stehlen oder Schaden anzurichten. Dazu gehören Ransomware, Spyware, Trojaner und Würmer. Malware nutzt häufig Sicherheitslücken (z.B. nicht gepatchte Software, veraltete Betriebssysteme oder gefährliche Browser-Plugins) aus, um Systeme zu infiltrieren. [BSI, 2025h]
- Phishing:
Bei diesem Angriff wird häufig Social Engineering mit technischen Tricks wie der Nachahmung einer echten Webseite oder gefälschten E-Mail-Absendern kombiniert,

um Benutzer zur Preisgabe sensibler Informationen zu verleiten, indem sie auf gefälschte Webseiten gelockt oder zum Öffnen schädlicher Anhänge verleitet werden. [BSI, 2025i]

- Drive-by Download:
Schadsoftware wird beim Besuch einer von Angreifern manipulierten Webseite auf das System geladen. Dies kann ohne aktive Zustimmung oder Kenntnis des Nutzers geschehen, oft ausgelöst durch Sicherheitslücken im Browser. [Oodrive, 2025] [BSI, 2025c]
- SQL-Injection:
Angriffe auf relationale Datenbanksysteme, bei denen Angreifer durch das Einschleusen von SQL-Befehlen unberechtigten Zugriff auf sensible Daten erlangen können. [BSI, 2025j]
- Man-in-the-Middle-Techniken:
Angreifer schalten sich zwischen die Kommunikation des Nutzers und der vermeintlich legitimen Website, um die eingegebenen Daten abzufangen, zu speichern oder zu manipulieren. Dies geschieht häufig durch Techniken wie IP-Spoofing oder Session-Hijacking, um unbemerkt sensible Informationen zu stehlen. [Oodrive, 2025]
- Abhören:
Beim passiven Abhören werden Kommunikationsdaten abgefangen, beim aktiven Abhören wird die Kommunikation zusätzlich manipuliert, indem sich der Angreifer als eine der Parteien ausgibt. [BSI, 2025f]

Nachdem die Absichten und die mögliche Ausführung eines Cyber-Angriffs bekannt sind, kann darauf auch reagiert werden. Unternehmen können definieren, warum sie sich mit Cybersicherheit befassen müssen. Dazu gehören die Vermeidung finanzieller Verluste, der Schutz des Rufs und des Kundenvertrauens, die Gewährleistung der Geschäftskontinuität und der Schutz geistigen Eigentums. [Pufahl et al., 2024] Zusammen mit den oben genannten Erwartungen ergeben sich daraus die Ziele der Cybersicherheit: [EU, 2025b]

- Vertraulichkeit: Zur Wahrung der Vertraulichkeit ist dafür Sorge zu tragen, dass nur Befugte personenbezogene Daten zur Kenntnis nehmen können.
- Integrität: Zur Wahrnehmung der Integrität ist sicherzustellen, dass personenbezogene Daten während der Verarbeitung und Nutzung unversehrt, vollständig und widerspruchsfrei bleiben.

- Verfügbarkeit: Zur Wahrung der Verfügbarkeit ist sicherzustellen, dass personenbezogene Daten zeitgerecht zur Verfügung stehen und ordnungsgemäß verarbeitet werden können.
- Authentizität: Zur Wahrung der Authentizität ist sicherzustellen, dass die Urheberschaft personenbezogener Daten jederzeit festgestellt werden kann.
- Revisionsfähigkeit: Es ist sicherzustellen, dass festgestellt werden kann, wer wann welche personenbezogenen Daten in welcher Weise verarbeitet hat.
- Transparenz: Zur Wahrung der Transparenz ist sicherzustellen, dass die Verfahrenswesen bei der Verarbeitung personenbezogener Daten vollständig, aktuell und in einer Weise dokumentiert sind, dass sie in zumutbarer Zeit nachvollzogen werden kann.

Für Software als Medizinprodukt spielen diese Ziele eine besonders wichtige Rolle. Ziel ist es, physische und psychische Schäden von Patienten, Anwendern und Dritten abzuwenden. [Johner Institut, 2025] Es gibt zahlreiche Möglichkeiten, die genannten Angriffsmethoden und Ziele der Cybersicherheit in die Praxis umzusetzen. Einen Überblick über mögliche Methoden und Maßnahmen zur Umsetzung von Cybersicherheit geben Darms, Haßfeld und Fedtke (2019). Die folgende Liste ist eine Zusammenfassung der genannten Aspekte bei der Umsetzung von Cybersicherheit:

- Schutz des Netzwerks:
 - Firewalls: Schutz vor unerwünschten Zugriffen auf das Netzwerk.
 - Intrusion Detection and Prevention Systeme: Erkennen und Blockieren verdächtiger Aktivitäten im Netzwerk.
 - Network Access Control: Regelung des Netzwerkzugangs auf Basis definierter Richtlinien.
- Zugangskontrolle und Authentifizierung:
 - Identitäts- und Zugriffsmanagement: Verwaltung von Benutzeridentitäten und Zugriffsrechten.
 - Authentifizierungsmethoden: Einsatz von Hardware-Token, biometrischen Verfahren, Passwörtern und Chipkarten/Smartcards.

- Autorisierung: Sicherstellen, dass Benutzer nur auf die Ressourcen zugreifen können, für die sie berechtigt sind.
- Anwendungs- und Datensicherheit:
 - Sandboxing: Isolierung von Anwendungen, um unbefugten Zugriff und Beschädigung des Systems zu verhindern.
 - Application Management: Kontrolle der Installation und Nutzung von Software.
 - Data Leakage and Loss Prevention: Schutz sensibler Daten vor unbefugter Offenlegung, Diebstahl und Verlust.
 - Verschlüsselung: Schutz sensibler Daten sowohl bei der Speicherung als auch bei der Übertragung durch Technologien wie VPNs, SSL/TLS-Zertifikate oder verschlüsselte Daten.
- Monitoring und Incident Management:
 - Security Information and Event Management: Zentrale Analyse und Verwaltung von sicherheitsrelevanten Informationen und Ereignissen.
 - Interface Monitoring: Schutz vor unbefugter Nutzung von z.B. USB, Bluetooth.
- System- und Schwachstellen-Management:
 - Zentrales Schwachstellen- und Patch-Management: Sicherstellung der Systemintegrität durch regelmäßige Updates.
 - Virtualisierung: Isolierung von Anwendungen zur Minimierung von Sicherheitsrisiken.
- Physische Sicherheit und Ausfallschutz:
 - Physische Sicherheitsmaßnahmen: Implementierung von Zugangskontrollen und Videoüberwachung.
 - Ausfallschutz: Sicherstellung der Verfügbarkeit durch unterbrechungsfreie Stromversorgung und Backup-Netzzugänge.

3. Methodik

Viele der in der Arbeit genannten Maßnahmen und Möglichkeiten sind nicht auf jede Software oder Infrastruktur anwendbar. Beispielsweise kann die als SaMD zugelassene Software OnyxCeph^{3TM} (siehe Abschnitt 1) keine physischen Sicherheitsmaßnahmen umsetzen, da dies in der Verantwortung des Kunden (der Praxis) liegt.

Ziel des folgenden Kapitels Methodik ist es daher, anhand der in den Grundlagen definierten gesetzlichen Regelungen für Software als Medizinprodukt eine Zuordnung zu den Methoden und Maßnahmen der Cybersicherheit zu finden.

3.1. Ermittlung von Cybersicherheitsanforderungen für OnyxCeph^{3TM}

In diesem Abschnitt werden Anforderungen an die Software OnyxCeph^{3TM} definiert, welche die Cybersicherheit erhöhen und aus den regulatorischen Anforderungen abgeleitet werden. Damit entsteht eine Basis für die Auswahl geeigneter Schutzmaßnahmen. Die aus den regulatorischen Anforderungen abgeleiteten konkreten Anforderungen (siehe Anhang A.2) sind in Tabelle 3.1 zusammengefasst und nach Herkunft, Ziel der Anforderung und den möglichen Schutzmaßnahmen gegliedert.

Tabelle 3.1.: Ermittlung von Cybersicherheitsmaßnahmen im Fall OnyxCeph^{3TM}

Herkunft	Ziel der Anforderung	Mögliche Sicherheitsmaßnahmen
DSGVO	Gewährleistung der Sicherheit personenbezogener Daten und Ermöglichung der Rückverfolgbarkeit der Datenverarbeitung. [EU, 2025b]	- Sicherstellen, dass nur autorisierte Benutzer Zugriff auf die Daten haben - Verschlüsselung der Daten zur Gewährleistung der Vertraulichkeit - Prozessisolierung zum Schutz der Datenintegrität
ISO14971	Minimierung der mit OnyxCeph ^{3TM} verbundenen Risiken durch Integration geeigneter Schutzmaßnahmen. [DIN, 2019]	- Durchführung regelmäßiger Risikoanalysen zur Identifizierung potenzieller Bedrohungen - Implementierung von Designstrategien zur Verbesserung der Sicherheit
ISO27001	Beschränkung des Zugriffs auf Informationen, Einhaltung von Datenschutzanforderungen und Schutz vor Datenlecks. [DIN, 2024]	- Durchführung von Identitätsprüfungen zur Zugriffskontrolle - Einsatz von Verschlüsselungstechniken, um die Vertraulichkeit der Daten zu sichern - Isolierung und Schutz von Daten und Prozessen vor unbefugtem Zugriff - Regelmäßige Analysen zur Aufdeckung von Schwachstellen
ISO13485	Verhinderung von Nichtkonformitäten und Umsetzung präventiver Maßnahmen gegen mögliche Probleme. [DIN, 2021]	- Absicherung der Prozesskommunikation und Schutz vor unberechtigtem Zugriff - Datenverschlüsselung zum Schutz der Vertraulichkeit - Durchführung regelmäßiger Kontrollen zur Sicherstellung der Konformität - Einsatz von Methoden zur Sicherung der Datenintegrität

Die in der Tabelle aufgeführten möglichen Schutzmaßnahmen dienen der Ausarbeitung spezifischer Schutzmaßnahmen. Der folgende Abschnitt beschäftigt sich daher mit den konkreten Ansätzen, die OnyxCeph^{3TM} nutzen kann, um diese Anforderungen zu erfüllen und die Cybersicherheit insgesamt zu verbessern.

3.2. Ansätze für Schutzmaßnahmen

Die aufgeführten Möglichkeiten zur Verbesserung der Cybersicherheit (siehe Tabelle 3.1) beschreiben lediglich Konzepte, die in der Praxis unterschiedlich umgesetzt werden können. Dieser Abschnitt befasst sich demnach mit der Frage, wie OnyxCeph^{3TM} die Sicherheit seiner Medizinproduktesoftware z.B. durch Verschlüsselung oder Authentifizierung erhöhen kann.

Verschlüsselung

Verschlüsselung ist eine seit der Antike bekannte Methode, um zu verhindern, dass unbefugte Dritte eine Nachricht zwischen Absender und Empfänger lesen können.

[Lenhard, 2020] Schon Julius Cäsar (100-44 v. Chr.) nutzte einfache Verschlüsselungsmethoden, um Kommunikationswege zu sichern. Der heute bekannte Cäsar-Code verschlüsselte militärische Nachrichten, indem er Zeichen durch Chiffren aus dem gleichen Alphabet ersetzte. [Ernst et al., 2023] Aus diesem Beispiel lassen sich wichtige Faktoren zur Umsetzung für OnyxCeph^{3TM} ableiten:

- Nachricht: sensible Information, die geschützt werden muss
- Verschlüsselungsmethode: stellt sicher, dass nur befugte Personen Zugriff auf die Nachricht haben
- Kommunikationsweg: der Weg, auf dem die Nachricht den Empfänger erreicht

OnyxCeph^{3TM} benötigt keine Kommunikation mit anderen Netzwerken. Folglich ist keine Datenverbindung nötig und damit auch kein Nachrichtenaustausch, sodass hier das zu schützende Objekt neu definiert werden muss. Fox (2009) bietet in diesem Zusammenhang eine mögliche Lösung an: „Eine weitere effektive Maßnahme ist die Verschlüsselung der auf dem System gespeicherten Daten.“ [Fox, 2009] OnyxCeph^{3TM} speichert zum Beispiel Patientenbilder temporär auf dem System, eine Datei, die als zu schützendes Objekt betrachtet wird. Die Verschlüsselung temporärer Daten ist demnach eine Möglichkeit, die Cybersicherheit sinnvoll zu erhöhen.

Authentifizierung

Authentifizierung wird oftmals mit der traditionellen Kombination aus Benutzername und Passwort assoziiert. Ziel dieser Kombination ist es, Vertraulichkeit zu gewährleisten. Dies wird erreicht, indem der Zugriff nur für authentische Benutzer möglich ist. Authentisch ist, wer seine Echtheit und Glaubwürdigkeit durch eine eindeutige Identität nachweisen kann. [Eichstädt and Spieker, 2024] Mitarbeitergespräche haben jedoch ergeben, dass die Authentifizierung mit Benutzername und Passwort bereits implementiert ist. Daraus folgt, dass fortschrittlichere Formen der Authentifizierung erforderlich sind. Eine Möglichkeit bietet Microsoft mit der Integration von Active Directory (AD). Active Directory bietet eine zentrale Plattform für die Verwaltung von Identitäten und Netzwerkressourcen. Mit Hilfe eines hierarchischen Modells, bestehend aus: [Microsoft, 2025p]

- Wäldern (engl. Forests): oberste Ebene der logischen Struktur, die eine Sammlung von einer oder mehreren Domänen mit einer gemeinsamen Konfiguration darstellt
- Domänen: logische Gruppierung von Netzwerkressourcen, die eine gemeinsame Datenbank (den Verzeichnisdienst) nutzen
- Organisationseinheiten: Container zur Strukturierung und Verwaltung von Objekten innerhalb einer Domäne

Ziel der Active Directory Domain Services ist es, die Verwaltung und Kontrolle von Zugriffsrechten in einer Netzwerkumgebung zu unterstützen und damit die Cybersicherheit zu erhöhen.

Eine weitere Möglichkeit ist die Verwendung von Hardware-Tokens, die als physische Sicherheitsgeräte dienen, z. B. in Form eines USB-Sticks. Diese Tokens stellen eine zusätzliche Sicherheitsebene dar. Ihr notwendiger physischer Besitz erschwert es Angreifern, den Token nachzubilden oder zu stehlen. Hardware-Token sind besonders effektiv, da der für den Zugriff auf Systeme oder Anwendungen benötigte Schlüssel verschlüsselt gespeichert wird. Ein Vertreter der Hardware-Token ist der FIDO2-Standard. FIDO2 (Fast IDentity Online 2) verwendet das WebAuthn-Protokoll zusammen mit einem FIDO2-Authentifikator, um eine passwortlose Authentifizierung zu ermöglichen. Der Authentifikator (z.B. ein USB-Stick) kommuniziert sicher über einen Client, oft ein Webbrowser, mit einem WebAuthn-Server. [FIDO, 2025] Dadurch kann OnyxCeph^{3TM} das Risiko von Phishing-Angriffen und anderen Online-Bedrohungen verringern, da der physische Besitz des Tokens für die Authentifizierung erforderlich ist.

Kapselung externer Bibliotheken

„Man muss das Rad nicht neu erfinden, sondern damit fahren.“ [Alfes et al., 2018] Ein Leitsatz, den sich auch die Softwareentwicklung zunutze macht. Der Einsatz externer Softwarelösungen bietet Entwicklern die Möglichkeit, Entwicklungszeiten und -kosten zu reduzieren und zusätzlich den aktuellen Stand der Technik zu nutzen. So kann es sinnvoll sein, benötigte Funktionalitäten durch den Kauf oder die Lizenzierung von Drittanbietern bereitzustellen. Dieser Ansatz ist jedoch mit Risiken verbunden, insbesondere wenn eine starke Abhängigkeit zu externen Lösungen besteht. Fehlerhafte und veraltete Fremdkomponenten können die Sicherheit und Funktionalität des Produkts beeinträchtigen. Um diesem Risiko entgegenzuwirken, bietet sich die Kapselung externer Komponenten an. Dabei wird die externe Komponente in einen eigenen Prozess gekapselt und der Kommunikationskanal zwischen Hauptprozess und Drittprozess klar definiert. Hierdurch werden direkte Abhängigkeiten minimiert und ein Austausch von Fremdkomponenten in der Zukunft ermöglicht. [Demant, 2018]

Ein weiterer Schritt zur Verbesserung der Systemsicherheit ist die Unterstützung der Prozessisolierung durch Sandboxing-Techniken. Sandboxing isoliert Prozesse in einer kontrollierten Umgebung, so dass Fehler oder Sicherheitslücken in einer Komponente nicht das gesamte System gefährden können. Diese Methode bietet eine effektive Kontrolle über den Zugriff auf kritische Ressourcen und reduziert das Risiko von Sicherheitsproblemen durch fehlerhafte oder böswillige Funktionalität von Drittanbietern. [Google, 2025]

Anonymisierung von Daten

Moderne Technologien wie Big Data, künstliche Intelligenz und Cloud Computing ermöglichen die Verarbeitung und Analyse großer Datenmengen. Diese Datenmengen bergen jedoch die Gefahr des Datendiebstahls. Um diesem Risiko entgegenzuwirken, ist die Anonymisierung von Daten eine relevante Maßnahme. [Liebe et al., 2024] Grundlegende Anonymisierungstechniken sind Generalisierung, Randomisierung und Aggregation.

Generalisierung ist ein Prozess, bei dem Daten auf eine allgemeinere Ebene transformiert werden. Anstatt genaue Angaben zu machen, wie z.B. ein bestimmtes Alter, können Daten zu größeren Kategorien zusammengefasst werden, wie z.B. Altersgruppen (20-30 Jahre). Durch die Verallgemeinerung wird es schwieriger, einzelne Personen zu identifizieren, wodurch die Privatsphäre geschützt wird. Ein weiterer Ansatz zum Schutz der Privatsphäre

ist die Randomisierung. Randomisierung bezieht sich auf Methoden, die Daten nach dem Zufallsprinzip verändern. Ziel ist es, die Verbindung zwischen den Daten und einzelnen Personen zu verwischen. Dies wird durch Techniken wie stochastische Überlagerung und statistische Verzerrung (Rauschen) erreicht. Bei der stochastischen Überlagerung werden Datenattribute zufällig verändert oder vertauscht, während bei der Verzerrung zufällige Störungen hinzugefügt werden, um den genauen Wert eines Attributs nicht bestimmbar zu machen. Die Aggregation bezeichnet schließlich den Prozess der Umwandlung einzelner Datensätze in aggregierte Daten. Statt spezifischer Informationen über einzelne Personen liefert die Aggregation Durchschnittswerte oder Summen, die keine individualisierbaren Daten mehr erkennen lassen. Diese Technik ermöglicht es, wertvolle Erkenntnisse zu gewinnen und gleichzeitig die Privatsphäre zu schützen, indem datengestützte Beurteilungen allgemeiner und weniger persönlich werden. [Krebs and Hagenweiler, 2022]

Vertrauenswürdige Betriebssysteme

Sicherheitslücken gar nicht erst entstehen zu lassen, ist ein Ziel der Cybersicherheit. Vertrauenswürdige Betriebssysteme (engl. Trusted Computing) setzen genau hier an, indem sie eine manipulationssichere und verifizierbare Systemumgebung schaffen. [Müller, 2008] Trusted Computing ist eine Technologie, die spezifische Hardware wie das Trusted Platform Module (TPM) verwendet. Das TPM dient als Chip, welcher Verschlüsselungsfunktionen bereitstellt und die Integrität von Soft- und Hardwarekomponenten durch die Bildung von Prüfsummen und deren Vergleich mit Referenzwerten sicherstellt. Dabei geht es nicht um eine formale Verifikation aller Softwarekomponenten, sondern um die Sicherstellung eines vertrauenswürdigen Systemzustandes. [Altenhofer and Studer, 2011] Die Definition von Trusted Computing beinhaltet jedoch unterschiedliche Perspektiven. So beschreibt Mason (2005) Trusted Computing als die Erwartung, dass ein System gemäß seiner Spezifikation funktioniert, während Schneider et al. (1999) die Zusicherung der korrekten Funktionsweise als zentral ansehen. [Mason, 2005][Schneider et al., 1999] Diese Unterschiede verdeutlichen die Vielseitigkeit von Trusted Computing, weshalb im Folgenden eine Liste von Zielen aufgeführt wird, die mit Trusted Computing verfolgt werden:

- Schaffung einer sicheren Umgebung in der Hardware als Fundament für die gesamte Sicherheitsarchitektur genutzt wird
- Sicherstellung der Systemintegrität durch verknüpfte, vertrauenswürdige Komponenten

- Überwachung und Absicherung durch gespeicherte Prüfsummen
- Schutz der Systemprozesse unabhängig voneinander, um Manipulationen zu verhindern

Darüber hinaus schützt Trusted Computing die Kommunikation zwischen Ein- und Ausgabegeräten sowie dem System mithilfe erweiterter Funktionen und stellt sicher, dass sicherheitsrelevante Dialoge nicht durch bösartige Software initiiert werden können.

[Müller, 2008] Trusted Computing bietet somit eine ganzheitliche Lösung, die sowohl die Sicherheit als auch das Vertrauen in IT-Systeme erhöht.

Verifizierung mit Werkzeugen

Um die Sicherheit von Softwaresystemen zu gewährleisten, ist neben der Anwendung von Schutzmaßnahmen auch das frühzeitige Erkennen von Sicherheitslücken notwendig. Ein Ansatz hierfür ist die Verwendung einer Softwarestückliste (SBOM). Die SBOM bietet einen systematischen Überblick über die Softwarekomponenten und deren Abhängigkeiten. Diese Übersicht wird durch folgende Architektur erreicht: [Dehning, 2024]

- Strukturierte und maschinenlesbare Dokumente, Dateien oder eine Datenbank
- Eine detaillierte Aufschlüsselung aller Komponenten
- Die Darstellung aller Abhängigkeiten einer Softwareanwendung

Die daraus resultierende Transparenz erleichtert nicht nur die Verwaltung der Komponenten, sondern ermöglicht auch die Verifizierung der SBOM. [Nehrke, 2023] Durch den automatisierten Abgleich der SBOM mit bekannten Schwachstellendatenbanken wie Common Vulnerabilities and Exposures (CVE) können potenzielle Risiken schnell identifiziert werden. CVE bietet eine Sammlung von derzeit 271.343 bekannten Cyberschwachstellen und ermöglicht es Entwicklern, gezielt auf diese Schwachstellen zu reagieren. [CVE, 2025]

Ein weiterer Verifikationsansatz sind Penetrationstests als Verfahren zur Überprüfung der Systemstabilität. Bei diesen Tests werden gezielte Angriffe auf die Software simuliert, um potenzielle Schwachstellen aufzudecken, die sonst unentdeckt bleiben würden. Durch die Realitätsnähe werden nicht nur Sicherheitslücken aufgedeckt, sondern auch die Reaktionsfähigkeit des Systems auf Bedrohungen getestet. Mit den gewonnenen Erkenntnissen kann OnyxCeph^{3TM} nicht nur seine Sicherheitsstrategien optimieren, sondern auch gezielt Anpassungen vornehmen, um sich vor neuen Bedrohungen zu schützen. Darüber hinaus

liefern die Ergebnisse eine Entscheidungsgrundlage für notwendige Investitionen in die Cybersicherheit, indem sie Verbesserungsbereiche aufzeigen. [Rominger et al., 2003]

3.3. Auswahl von Schutzmaßnahmen

In diesem Abschnitt wird der Entscheidungsprozess beschrieben, der zur Auswahl der spezifischen Schutzmaßnahmen geführt hat. Die Entscheidungen wurden auf der Basis von Mitarbeitergesprächen getroffen. Dabei wurden die einzelnen Schutzmaßnahmen nach folgenden Kriterien analysiert:

- Umsetzbarkeit der Maßnahme
- Kosten-Nutzen-Verhältnis
- Wirksamkeit, Ziel ein breites Spektrum an Angriffsmöglichkeiten abzudecken
- Kompatibilität, reibungslose Einführung der Maßnahme in die bestehende IT-Infrastruktur

Ausschluss bestimmter Schutzmaßnahmen

Bei der Untersuchung von Trusted Computing wurde festgestellt, dass diese Technologie eine potenzielle Angriffsfläche bietet. Sicherheitsforscher haben Schwachstellen in TPM 2.0-Chips entdeckt, die es einem Angreifer ermöglichen, an vertrauliche Daten wie kryptographische Schlüssel zu gelangen oder Benutzerrechte unberechtigt zu erweitern und damit das gesamte System zu kompromittieren. [Pohlmann, 2024] [Beiersmann, 2025] Aus diesem Grund wird auf die Implementierung von Trusted Computing verzichtet.

Es wurde auch deutlich, dass die Anonymisierung der Daten nicht im Fokus der Entwickler steht. Der Grund dafür ist, dass die Praxen selbst über die Anonymisierung ihrer Patientendaten entscheiden. Der Schwerpunkt liegt vielmehr auf den Sicherheitsmaßnahmen im direkten Umfeld der Softwareentwicklung.

Aus einem ähnlichen Grund wird auf die Verwendung eines zentralen Verzeichnisdienstes verzichtet. Die Einrichtung einer Active Directory liegt in der Verantwortung der Praxis und könnte daher nur für die Infrastruktur der Entwickler genutzt werden. Aufgrund dieser fehlenden Reichweite wird diese Schutzmaßnahme nicht weiter betrachtet.

Die Verifikation durch Softwarestücklisten ist zwar sinnvoll, wird hier jedoch nicht weiter vertieft, da die Verwaltung von Schwachstellen in der Software OnyxCeph^{3TM} mithilfe von SBOMs und CVE-Datenbanken bereits umgesetzt wird. Der Fokus dieser Masterarbeit liegt deshalb auf neuen Ansätzen.

Schließlich erfordert die effektive Durchführung von Penetrationstests spezifische Kenntnisse über die Funktionsweise der internen Softwarekomponenten. Da die Aneignung dieser Kenntnisse jedoch zu ressourcenintensiv ist, wurde beschlossen, das Thema nicht weiterzuverfolgen.

Ausgewählte Schutzmaßnahmen

Bei der Verschlüsselung wird der Schutz der Vertraulichkeit der sensiblen Patientendaten als zentral angesehen. Dementsprechend werden temporäre Daten, die unverschlüsselt auf dem System abgelegt werden, als Risiko betrachtet. Da diese Daten häufig auch sensible Informationen enthalten, wird eine Verschlüsselung der temporären Daten als notwendig erachtet.

Prozessisolation durch Kapselung und Sandboxing sind Strategien zur Sicherstellung der Systemintegrität. Insbesondere wenn es um den Umgang mit Komponenten von Drittanbietern geht. Die Minimierung von Abhängigkeiten und die Reduzierung des Risikos durch unsichere Software stehen im Vordergrund, weshalb eine prototypische Umsetzung beabsichtigt ist.

Erweiterte Authentifizierungsmethoden sind vielversprechend, um die Sicherheit sensibler Patientendaten zu erhöhen und einen unbefugten Zugriff zu verhindern. Daher soll untersucht werden, wie sich die bestehende Authentifizierung mit Benutzername und Passwort erweitern lässt, um insbesondere gegen Phishing- und Man-in-the-Middle-Angriffe besser abgesichert zu sein.

3.4. Planung der Auswertung

Cybersicherheitsmaßnahmen haben nicht nur Vorteile für das Gesamtsystem. Deshalb sollten bereits bei ihrer Planung die Kosten-Nutzen-Aspekte besonders beachtet und bei der Bewertung auch die Nachteile berücksichtigt werden. [Pohlmann, 2022] Entstehende Kosten durch Leistungsverluste im System oder notwendige Vorkenntnisse für die Implementierung können ein entscheidender Faktor für die Auswahl einer Schutzmaßnahme sein. Aus diesem Grund soll eine Gegenüberstellung der spezifischen Schutzmaßnahmen in tabellarischer Form erfolgen. Durch die Zuordnung von Zahlenwerten zu definierten Faktoren soll ein Mehrwert für die Entscheidung über eine Schutzmaßnahme geschaffen werden. Die Auswahl der Faktoren zielt darauf ab, aufzuzeigen, welche Maßnahme unter welchen Bedingungen entwickelt werden kann. Dadurch sollen die geringsten Entwicklungskosten bei gleichzeitig höchstem potenziellen Schutz und minimalem Verlust der Programmlaufzeit erreicht werden. Um das Gleichgewicht zwischen diesen Zielen zu finden und somit die Entscheidung für eine Schutzmaßnahme zu unterstützen, werden die folgenden Faktoren ausgewählt und berücksichtigt:

- **Erforderliche Vorkenntnisse:** Dieser Faktor beschreibt das Maß der benötigten fachlichen Kenntnisse für die erfolgreiche Umsetzung der Maßnahme.
- **Zeitaufwand und Schwierigkeit der Implementierung:** Diese Faktoren umfassen sowohl die zeitliche Dauer als auch die Komplexität der Implementierung.
- **Anfallende Kosten:** Dieser Faktor gibt an, ob und in welchem Ausmaß für die Implementierung der Maßnahme zusätzliche finanzielle Ressourcen benötigt werden.
- **Zusätzliche Programmlaufzeit:** Dieser Faktor behandelt die durch die Implementierung der Schutzmaßnahme bedingte Verlängerung der Programmlaufzeit.
- **Nutzen der Maßnahme:** Dieser Faktor bewertet den Vorteil der Maßnahme, Angriffsmöglichkeiten zu verhindern oder abzuschwächen.

Um diese Faktoren quantitativ bewertbar zu machen, erfolgt eine Bewertung mittels eines Punktesystems auf einer Skala von 1 bis 10, wobei der Wert 1 für den minimalen und der Wert 10 für den maximalen Einfluss steht. Die Bewertung selbst erfolgt anhand empirischer Messungen, die einem anwendungsorientierten Forschungsansatz folgen. Hierbei kommen realistische Szenarien und Testbedingungen zum Einsatz, um die ausgewählten Schutzmaßnahmen zu evaluieren.

4. Umsetzung

„Wer hohe Türme bauen will, muss lange beim Fundament verweilen.“ [Sänger, 2010]
Dieses Zitat von Josef Anton Bruckner verdeutlicht die Wichtigkeit der notwendigen Vorbereitung, um Schutzmaßnahmen gewissenhaft umzusetzen. In diesem Kapitel wird sowohl der theoretische Hintergrund als auch die prototypische Umsetzung der im vorangegangenen Kapitel ausgewählten Schutzmaßnahmen für OnyxCeph^{3TM} beschrieben. Das Fundament, auf dem die Maßnahmen aufbauen, wurde durch eine Analyse der Anforderungen an die Cybersicherheit sowie durch Mitarbeitergespräche erarbeitet. Ziel ist es, die Maßnahmen prototypisch umzusetzen, um die Erhöhung der Cybersicherheit zu demonstrieren. Die Entwicklung der Prototypen erfolgt in der gleichen Entwicklungsumgebung wie OnyxCeph^{3TM}. Zum Einsatz kommt Windows als Betriebssystem mit Embarcadero als IDE unter Verwendung der Programmiersprache C++ und der Version zwölf des C++ Builders. In den folgenden Abschnitten wird demnach beschrieben, wie Verschlüsselung, Authentifizierung und weitere ausgewählte Sicherheitsstrategien in der Praxis umgesetzt werden.

4.1. Verschlüsselung

Für die Umsetzung von Verschlüsselungsmaßnahmen ist es zu Beginn notwendig, ein geeignetes Szenario zu definieren. Anhand dieses Szenarios sollen die praktischen Schritte und Überlegungen dargestellt werden. Als Beispiel soll die Speicherung eines Bildes auf dem System dienen, wobei davon ausgegangen wird, dass das Bild sensible Patienteninformationen enthält. Bevor jedoch auf die Verschlüsselung der Bilddaten eingegangen werden kann, muss geklärt werden, in welchem Format die Bilder vorliegen und wie die Daten ausgelesen und auf dem Rechner gespeichert werden.

4.1.1. Windows Bitmap

Windows Bitmap (.bmp) ist ein historisches, aber immer noch verwendetes Bildformat zum Speichern von Bilddaten. [Bourke, 1998] Das geräteunabhängige Bildformat besteht typischerweise aus einem Header, der grundlegende Informationen wie Breite, Höhe und Anzahl der Bits pro Pixel enthält, gefolgt von den Bilddaten. Der im Header definierte Offset gibt an, wo die Bilddaten in der Datei beginnen. Abhängig von der Anzahl der Bits pro Pixel kann eine Bitmap verschiedene Farbmodi haben. Bei einem 1-Bit-Bild entspricht dies Schwarz-Weiß, bei einem 24-Bit-Bild dem RGB-Farbmodell, bei dem jedes Pixel aus drei Bytes besteht, die die Intensität von Rot, Grün und Blau bestimmen. In diesem Fall handelt es sich um ein 24-Bit-Bild, dessen Struktur nicht komprimiert ist, was das Einlesen und Verarbeiten der Daten erleichtert. Die Werte werden in hexadezimaler Form angegeben, wobei 00_{hex} für die minimale Intensität und FF_{hex} für die maximale Intensität der jeweiligen Farbe steht. [Microsoft, 2025b] Zur Veranschaulichung zeigt die Abbildung 4.1 auf der einen Seite ein Array von Farbwerten und daneben das zugehörige Bild. Die verwendeten Farben sind wie folgt definiert:

- FFFFFFFF, entspricht Weiß
- FF0000, entspricht Rot
- 00FF00, entspricht Grün
- 0000FF, entspricht Blau

```

0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF
00FF00 FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF 00FF00
00FF00 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 00FF00
00FF00 FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF 00FF00
00FF00 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 00FF00
00FF00 FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF 00FF00
00FF00 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 00FF00
0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF

```



Quelle: [Microsoft, 2025b]

Abbildung 4.1.: Hexadezimale Darstellung eines Pixelbildes

4.1.2. Streaming-Klassen zum Speichern und Lesen von Bilddaten

In C++ werden Streams verwendet, um die Ein- und Ausgabe von Daten zu ermöglichen. Streams sind Teil der C++-Standardbibliothek und bieten eine einheitliche Schnittstelle für das sequentielle Lesen und Schreiben von Daten. [Microsoft, 2025i] Die Übertragung ist somit unabhängig vom Absender oder Empfänger der Daten. In unserem konkreten Anwendungsfall wird die Klasse `TFileStream` von Embarcadero verwendet, um mit Bilddaten zu arbeiten. [Embarcadero, 2025a] Diese Klasse erbt von `TStream` und behandelt Dateien als eine kontinuierliche Folge von Bytes, was das Lesen und Schreiben von Binärdaten vereinfacht. Die Klasse `TFileStream` übernimmt Aufgaben wie das Puffern von Daten und die Verwaltung des Dateizeigers, der bestimmt, wo in der Datei gelesen oder geschrieben werden soll. [Embarcadero, 2025b]

Bevor die Bilddaten über einen Stream in eine Datei geschrieben werden, ist es für weitere Funktionalitäten sinnvoll, die Bilddaten in einem Array zwischenspeichern. Wie in der folgenden Abbildung 4.2 dargestellt, ergibt sich die Größe des Arrays aus der Bildbreite, der Bildhöhe und der Anzahl der Farbkanäle (RGB entspricht 3 Farbkanälen):

```
int arraySize = width * height * 3
```

Abbildung 4.2.: Formel zur Bestimmung der Arraygröße

Um die Datenübertragung effizient zu gestalten, wird die Arraygröße in Abbildung 4.3 zusätzlich auf das nächste Vielfache der Blockgröße von 512 Byte angepasst. Dadurch werden die Datenblöcke an die Speicherstruktur angepasst, was die Effizienz der Datenverarbeitung erhöht. Die Wahl der Blockgröße von 512 basiert auf der typischen Sektorgröße von Festplatten. Diese Größe wurde als Standard festgelegt, da sie der physischen Sektorgröße auf den meisten Festplatten entspricht und somit die Leistung beim Lesen und Schreiben von Daten optimiert. [Zirkel, 2004]

```
int rest = arraySize % BlockSize;  
arraySize += BlockSize - rest;  
unsigned char* array = new unsigned char[arraySize];
```

Abbildung 4.3.: Code zur Anpassung der Arraygröße auf ein Vielfaches der Blockgröße

Die Bilddaten werden nun zeilenweise in das Array geladen, wodurch eine geordnete Speicherung der Pixelwerte gewährleistet wird. Sobald das Array korrekt angelegt ist, können die Bilddaten wie in Abbildung 4.4 dargestellt sequenziell in den Stream geschrieben werden.

```
stream->WriteBuffer((unsigned char*)&width, 4);
stream->WriteBuffer((unsigned char*)&height, 4);
for(int position = 0; position < arraySize; position += BlockSize) {
    stream->WriteBuffer(array + position, BlockSize);
}
```

Abbildung 4.4.: Datei mit Bilddaten unter Verwendung eines Streams erstellen

Der selbe Ansatz kann verwendet werden, um Bilddaten aus einer Datei auszulesen und das Bild daraus zu rekonstruieren. Durch die Anwendung der beschriebenen Methode erhält man eine klar strukturierte und interpretierbare Datei, die alle notwendigen Informationen enthält. Sie beginnt mit der Bildbreite und -höhe, gefolgt von den eigentlichen Bilddaten (siehe Auszug im Anhang A.3). Mit Hilfe der Datenstruktur kann die Datei und damit auch die Bilddaten verschlüsselt werden.

4.1.3. Grundlagen der Verschlüsselung

Bevor auf die Implementierung von Verschlüsselungsverfahren eingegangen wird, soll zunächst ein Exkurs in die Grundlagen der Verschlüsselungstechnik erfolgen. Dieser Abschnitt befasst sich mit der grundsätzlichen Funktionsweise von Verschlüsselung, der Bedeutung von Schlüsseln für die Verschlüsselung sowie den Unterschieden zwischen symmetrischen und asymmetrischen Verfahren. Wichtig für den Einsatz von Verschlüsselung im Datenschutz ist, dass der Personenbezug durch die Verschlüsselung nicht aufgehoben wird. Das bedeutet, dass auch verschlüsselte personenbezogene Daten weiterhin dem Datenschutz unterliegen. [Kneuper, 2021]

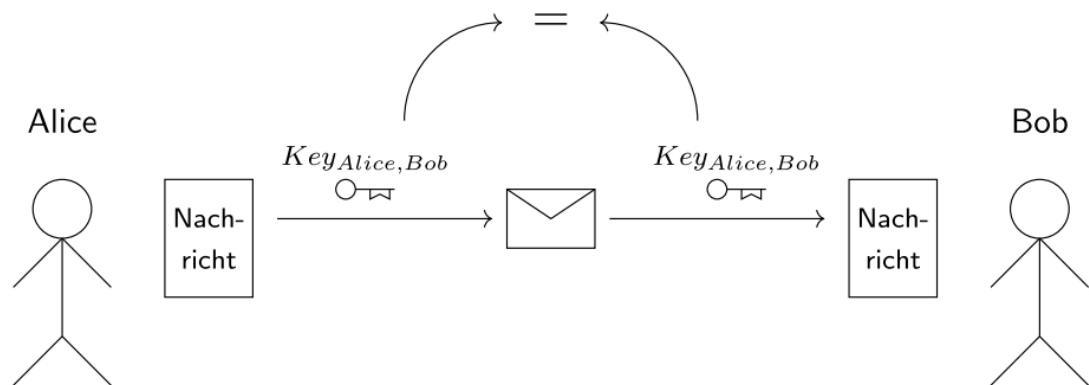
Verschlüsselung ist die Umwandlung von Klartext, also lesbaren Daten, in Geheimtext. Diese Umwandlung erfolgt in Abhängigkeit vom so genannten Schlüssel. Der Schlüssel ist entscheidend, da er sowohl für die Verschlüsselung als auch für die Entschlüsselung benötigt wird. Die Geheimhaltung des Schlüssels bestimmt die Sicherheit des Verschlüsselungsverfahrens. Ist die Geheimhaltung des Schlüssels gewährleistet, ist es für einen Angreifer nur begrenzt möglich, den ursprünglichen Klartext wieder umzuwandeln. Ausgenommen

ist das Erraten des richtigen Schlüssels, also ein Brute-Force-Angriff. Die Schlüssellänge variiert je nach Verschlüsselungsverfahren, weshalb zunächst auf die verschiedenen Verfahren eingegangen werden muss. [Meinel et al., 2014]

Symmetrische Verschlüsselung

Bei diesem Verfahren wird ein einziger Schlüssel für die Verschlüsselung und Entschlüsselung der Daten verwendet. Dieser Schlüssel ist für Sender und Empfänger identisch und muss zwischen den Kommunikationspartnern sicher ausgetauscht werden. Der Vorteil dieses Verfahrens liegt in der Effizienz ohne unvermeidbare Performanceverluste. Demgegenüber steht die Herausforderung der symmetrischen Verschlüsselung, diesen Schlüssel sicher auszutauschen. Ohne geeignete Sicherheitsmaßnahmen kann der Austausch anfällig für Abhörmethoden wie Man-in-the-Middle-Angriffe sein, bei denen ein Angreifer die Kommunikation abfangen und den Schlüssel ausspionieren kann, um die Nachricht zu entschlüsseln.

Zur Illustration der symmetrischen Verschlüsselung zeigt Abbildung 4.5, dass Alice eine geheime Nachricht an Bob senden möchte und dafür die symmetrische Verschlüsselung verwendet. Beide einigen sich vorher auf einen geheimen Schlüssel, mit dem sie die Nachricht verschlüsseln und später wieder entschlüsseln. Alice verschlüsselt also die Nachricht auf dem Kommunikationsweg, damit Bob die Nachricht mit dem vereinbarten Schlüssel entschlüsseln kann. Während früher typischerweise Algorithmen des Data Encryption Standard (DES) mit einer Schlüssellänge von 64 Bit zur Verschlüsselung der Nachricht verwendet wurden, ist dieser Standard mittlerweile veraltet und gilt als unsicher. [Meinel et al., 2014] Heute werden Algorithmen wie der Advanced Encryption Standard (AES), der von der amerikanischen Standardisierungsbehörde NIST in einem offenen Wettbewerbsverfahren als Standard ausgewählt wurde, mit Schlüssellängen von 128 Bit oder 256 Bit verwendet, um den Sicherheitsanforderungen weiterhin gerecht zu werden. [Kneuper, 2021]



Quelle: [Kneuper, 2021, S.184]

Abbildung 4.5.: Symmetrische Verschlüsselung

Asymmetrische Verschlüsselung

Die asymmetrische Verschlüsselung ist ein Verfahren, bei dem zwei verschiedene Schlüssel für die Verschlüsselung und Entschlüsselung zwischen zwei Kommunikationspartnern verwendet werden. Jeder Teilnehmer besitzt ein Schlüsselpaar, das aus einem öffentlichen Schlüssel (engl.: public key) und einem privaten Schlüssel (engl.: private key) besteht. Jeder dieser Schlüssel übernimmt eine eindeutige Aufgabe: [Kneuper, 2021]

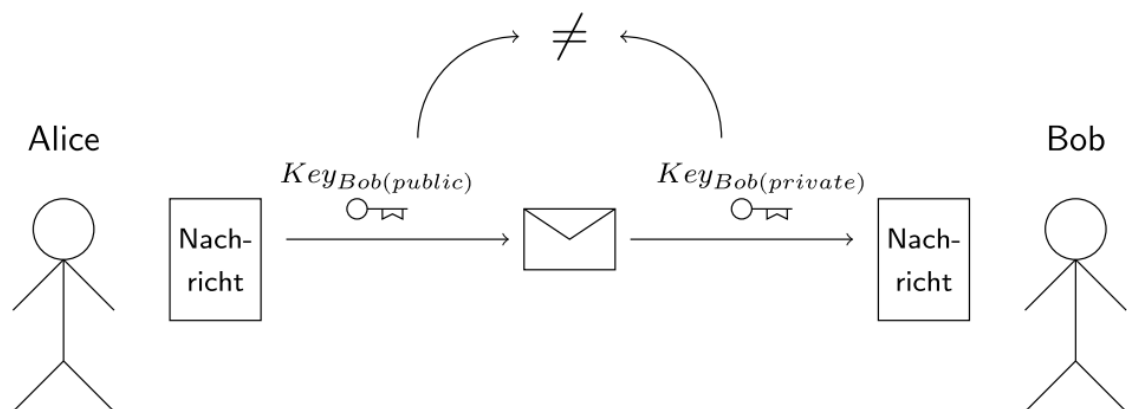
- Öffentlicher Schlüssel: kann frei verteilt werden und wird verwendet, um Nachrichten an den Besitzer des zugehörigen privaten Schlüssels zu verschlüsseln
- Privater Schlüssel: wird geheim gehalten und verwendet, um Nachrichten zu entschlüsseln, die mit dem zugehörigen öffentlichen Schlüssel verschlüsselt wurden

Ebenfalls grundlegend für die asymmetrische Verschlüsselung ist das Konzept der Einwegfunktionen. Dabei handelt es sich um mathematische Gleichungen, die relativ einfach zu berechnen sind, deren Umkehrung aber einen enormen Rechenaufwand erfordert. Eine gängige Einwegfunktion ist die Multiplikation zweier großer Primzahlen. Während das Produkt dieser beiden Zahlen einfach zu berechnen ist, ist es schwierig, dieses Produkt wieder in seine ursprünglichen Primzahlen zu zerlegen. Diese Eigenschaft macht sich der RSA-Algorithmus von Rivest, Shamir und Adleman zunutze. So kann aus dem Produkt zweier Primzahlen der öffentliche Schlüssel gebildet und frei geteilt werden, während die Primfaktoren geheim bleiben. [Buell, 2024] Im Zusammenhang mit dem RSA-Algorithmus wird typischerweise eine Schlüssellänge von mindestens 2048 Bit verwendet. Die zur Erzeugung des RSA-Schlüssels verwendeten Primzahlen werden in der Regel so gewählt, dass ihr

Produkt die gewünschte Schlüssellänge erreicht. Die Primzahlen sollten daher etwa 1024 Bit lang sein. Je größer die Primzahlen sind, desto schwieriger ist es, den öffentlichen Schlüssel durch Brute-Force-Angriffe in seine Faktoren zu zerlegen. [Barker et al., 2007]

Um die Funktionsweise der asymmetrischen Verschlüsselung weiter zu verdeutlichen, wird das Beispiel mit den beiden Kommunikationspartnern Alice und Bob in Abbildung 4.6 wieder aufgegriffen. In einer typischen Kommunikationssituation möchte Alice eine verschlüsselte Nachricht an Bob senden. Dazu verwendet Alice den öffentlichen Schlüssel von Bob, um die Nachricht zu verschlüsseln. Bob kann die Nachricht dann mit seinem privaten Schlüssel entschlüsseln. Umgekehrt kann Bob Nachrichten an Alice senden, die er mit Alices öffentlichem Schlüssel verschlüsselt und die nur Alice mit ihrem privaten Schlüssel entschlüsseln kann.

Dieser Ansatz ermöglicht eine sichere Kommunikation auch in unsicheren Netzwerken, ohne dass die privaten Schlüssel jemals ausgetauscht werden müssen. Weiterführend beschäftigen sich Cocks-Ellis-Williamson und Diffe-Hellman mit dem Algorithmus zum Schlüsselaustausch zwischen Kommunikationspartnern, die sich nicht kennen und daher keinen öffentlichen Schlüssel verwenden können. [Buell, 2024]



Quelle: [Kneuper, 2021, S.184]

Abbildung 4.6.: Asymmetrische Verschlüsselung

Sicherheit von Verschlüsselungsverfahren

Ein Grundsatz der Sicherheit von Verschlüsselungsverfahren ist das Kerckhoff Prinzip. Dieser Grundsatz beschreibt, dass die Sicherheit eines Verschlüsselungssystems nicht auf

der Geheimhaltung des verwendeten Algorithmus basieren sollte, sondern auf der Geheimhaltung des Schlüssels selbst. Dafür gibt es mehrere Gründe: [Kneuper, 2021]

- Verschlüsselungsalgorithmen bieten in der Regel viele Angriffspunkte, mehr als ein einzelner Schlüssel
- Wenn ein Algorithmus kompromittiert wird, ist es einfacher, einen Schlüssel zu ersetzen, als einen ganzen Algorithmus neu zu entwickeln
- Flexibilität bei der Kommunikation mit mehreren Parteien
- Für jeden Kommunikationspartner kann ein individueller Schlüssel generiert werden, was die Anpassungsfähigkeit des Systems erhöht

4.1.4. Implementierung der Datenverschlüsselung

Mit der notwendigen Vorbereitung kann in diesem Abschnitt die praktische Umsetzung von Verschlüsselungstechniken anhand eines Beispielprojektes untersucht werden. Ziel ist es, Bilddaten zu verschlüsseln und dabei ausgewählte Verschlüsselungsverfahren zu testen. Die folgenden Vorgehensweisen zeigen die Herausforderungen und Lösungsansätze beim Einsatz von Verschlüsselungsverfahren in der Praxis. Ausgangspunkt ist das in Abbildung 4.7 dargestellte Bild mit einer Auflösung von 5034 x 5269 Pixeln und einer Dateigröße von 75,8 MB. Die Verschlüsselung dieses Bildes dient als Konzeptnachweis für die Wirksamkeit und die Herausforderungen bei der Anwendung von Verschlüsselungstechniken.



Quelle: [Lukeman, 2025]

Abbildung 4.7.: Landschaftsbild im BMP-Format

XOR-Verschlüsselung

Die XOR-Verschlüsselung ist eine der grundlegenden Verschlüsselungsmethoden. Sie funktioniert, indem jedes Byte der Originaldaten mit einem Schlüsselbyte durch eine XOR-Operation verknüpft wird. Im Projekt werden die Bilddaten in Blöcken von 512 Byte verarbeitet. Diese Blockgröße, die bereits zur Optimierung der Lese- und Schreiboperationen erläutert wurde, hat noch einen zweiten Vorteil. Verschlüsselung und Entschlüsselung benötigen für eine sequentielle Abarbeitung die gleiche Ein- und Ausgabelänge, was durch die Aufteilung in 512-Byte-Blöcke ermöglicht wird. Der folgende Codeausschnitt 4.8 zeigt, wie die Verschlüsselung implementiert wurde. Wichtig bei der Schlüsselerzeugung ist, dass der Schlüssel Werte im gesamten Bereich von 00_{hex} bis FF_{hex} erzeugt, um die Zufälligkeit und Entropie des verschlüsselten Bildes zu erhöhen. Dadurch wird die Vorhersagbarkeit und folglich die Angreifbarkeit der Verschlüsselung reduziert. Es folgt die sequentielle Verschlüsselung der Bilddaten nach Blockgröße. Schließlich werden Breite und Höhe, gefolgt von den Bilddaten, in einem Stream auf das System geschrieben.

```
for (int i = 0; i < BlockSize; i++) {
    Key1[i] = (rand() % 255);
}
for (int count = 0; count < arrSize; count++) {
    arr[count] = arr[count] ^ Key1[keyCount++];
    if (keyCount == BlockSize)
        keyCount = 0;
}
stream->WriteBuffer((unsigned char*)&width, 4);
stream->WriteBuffer((unsigned char*)&height, 4);
for(int position = 0; position < arrSize; position+=BlockSize) {
    stream->WriteBuffer(arr+position, BlockSize);
}
```

Abbildung 4.8.: Implementierung der XOR-Verschlüsselung

Obwohl das Bild, wie in Abbildung 4.9 dargestellt, verschlüsselt auf dem System gespeichert wird und ohne Entschlüsselung für den Menschen nicht identifizierbar ist, weist die XOR-Verschlüsselung dennoch Sicherheitslücken auf: [Rimscha, 2018]

- **Strukturen:** Wenn Farbwerte im Originalbild häufig vorkommen, wird ein entsprechender Farbwert auch im kodierten Text häufig vorkommen. Diese Methode erhält also Strukturen des Originalbildes.

- Angriffspunkte: Eine bekannte Schwachstelle der XOR-Verschlüsselung ist die Known-Plaintext-Attacke. Kennt ein Angreifer nur eine Klartextstelle im Originalbild und deren verschlüsselte Form, kann er daraus den Schlüssel ableiten.

Diese Unsicherheiten zeigen, dass das Kerckhoff Prinzip in der Praxis nicht erfüllt ist, da der Schlüssel ableitbar ist und Strukturen der Nachricht im Chiffre erhalten bleiben.



Abbildung 4.9.: XOR-verschlüsseltes Landschaftsbild

Auswahl eines sicheren Verschlüsselungsverfahrens

Unter Berücksichtigung der oben genannten Schwächen ist eine Umstellung auf ein anerkanntes und sicheres Verschlüsselungsverfahren notwendig. In Frage kommen das asymmetrische Verschlüsselungsverfahren RSA und das symmetrische Verschlüsselungsverfahren AES. RSA bietet zwar Vorteile bei der sicheren Schlüsselverteilung, ist aber mit einem höheren Rechenaufwand verbunden. [Karpfinger and Kiechle, 2010] Da die Software OnyxCeph^{3TM} keine Internetverbindung voraussetzt, würden die Vorteile von RSA nicht optimal genutzt und die Performanceverluste überwiegen. Stattdessen wird die symmetrische AES-Verschlüsselung umgesetzt.

AES-Verschlüsselung

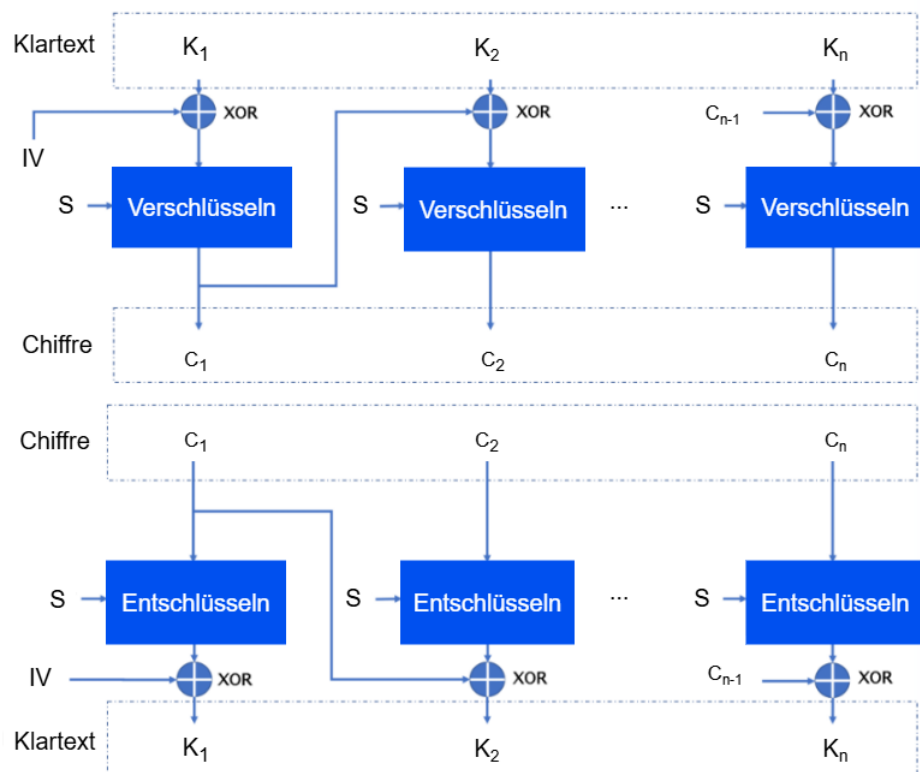
Die Implementierung komplexer Verschlüsselungsalgorithmen wie AES von Grund auf wird nicht angestrebt. Der Grund dafür ist, dass in der Kryptographie von der Entwicklung eigener Verschlüsselungsalgorithmen abgeraten wird. Dies schließt auch die Eigenimplementierung bestehender Algorithmen wie AES ein. [OWASP, 2025] Denn damit sind Herausforderungen und Risiken verbunden, insbesondere im Hinblick auf Fehleranfälligkeit und Sicherheit. Dies zeigt das Beispiel des Heartbleed-Bugs in OpenSSL im Jahr 2014, der eine Sicherheitslücke in TLS-verschlüsselten Verbindungen aufdeckte, die bei der Verarbeitung von Protokolldaten einen ungewollten Lesezugriff auf interne einfache Datenstrukturen im Speicher ermöglichte. [Kebschull, 2023] Allerdings profitieren solche Bibliotheken von großen Entwicklergemeinschaften, die im Gegensatz zu Eigenimplementierungen schnell auf Sicherheitslücken reagieren können. Daher wird im Folgenden die etablierte Bibliothek OpenSSL zur Implementierung der AES-Verschlüsselung verwendet.

OpenSSL bietet eine Reihe von Verschlüsselungsoptionen. Zu den unterstützten Betriebsarten gehört AES in verschiedenen Konfigurationen wie: [OpenSSL, 2025b]

- AES-256-CBC
- AES-256-CTR
- AES-256-ECB, und viele mehr

Diese Vielfalt ermöglicht es, ein breites Spektrum an Sicherheitsanforderungen abzudecken. Im Rahmen des vorliegenden Projekts wurde der Modus AES-256-CBC gewählt, da dieser nach aktuellen Standards zur Gewährleistung der Vertraulichkeit sensibler Daten eingesetzt wird. Im Gegensatz zu anderen Modi, die zusätzliche Funktionen zur Sicherstellung von Authentizität und Integrität bieten, ist CBC insbesondere dann sinnvoll, wenn eine sichere Verschlüsselung im Mittelpunkt steht. [BSI, 2025k] AES-256-CBC bezeichnet die AES-Verschlüsselung mit 256-Bit-Schlüssel und Blockverkettung (CBC). Bei der Blockverkettung wird jeder Block vor der Verschlüsselung mit dem vorhergehenden Chiffreblock verkettet. Dadurch wird sichergestellt, dass identische Klartextblöcke durch die Verkettung unterschiedliche Chiffreblöcke erzeugen, was die Sicherheit gegen Mustererkennung und die Vorhersagbarkeit erhöht. Ein wesentlicher Bestandteil dieses Verfahrens ist der Initialisierungsvektor (IV). Er sorgt dafür, dass identische Daten bei jeder Verschlüsselung zu unterschiedlichen Ergebnissen führen. Diese Zufälligkeit verleiht der Verschlüsselung eine nichtdeterministische Eigenschaft. Dabei wird jeder Klartextblock mit

dem vorhergehenden Chiffreblock durch eine XOR-Operation verknüpft, wodurch die Abhängigkeit zwischen den Blöcken weiter verstärkt wird. Der IV wird für den ersten Block verwendet, da vorher kein Chiffreblock zur Verkettung zur Verfügung steht. Die eigentliche Verschlüsselung jedes Blocks erfolgt dann durch symmetrische Verschlüsselung des verknüpften Blocks mit einem vorher festgelegten Schlüssel. [Alsman et al., 2023] Dieses Verfahren ist in Abbildung 4.10 veranschaulicht.



Quelle: [Alsman et al., 2023]

Abbildung 4.10.: AES-Verschlüsselung im CBC-Modus

Der Ausgangspunkt für die Implementierung der Verschlüsselung ist das Bild 4.7, dessen Daten in der Anwendung eingelesen und verarbeitet werden. Die Daten werden in Blöcke von jeweils 512 Byte unterteilt. Die vorliegende Blockgröße gewährleistet, dass für die Verschlüsselung und Entschlüsselung die gleiche Eingabe- und Ausgabelänge verwendet werden kann. Die Realisierung der Verschlüsselung erfolgt mittels einer Hilfsklasse, welche den Verschlüsselungsprozess steuert und die von OpenSSL bereitgestellten Funktionen nutzt. In Anbetracht dessen wird ein Verschlüsselungskontext generiert, um die Vorbereitung zu vervollständigen. Für die Realisierung der Implementierung wurde für den CBC-Modus von AES ein 256-Bit-Schlüssel in Kombination mit einem 128-Bit-Initialisierungsvektor verwendet. Im Folgenden werden die von OpenSSL bereitgestellten Funktionen hinsichtlich der Initialisierung, der Aktualisierung und des Abschlusses implementiert. [OpenSSL, 2025a]

Die Initialisierung der Verschlüsselung, wie in Abbildung 4.11 dargestellt, erfolgt durch die Erzeugung eines Verschlüsselungskontexts, welcher durch die Funktion `EVP_CIPHER_CTX_new` erzeugt wird. Im darauffolgenden Schritt wird der Verschlüsselungsprozess durch die Funktion `EVP_EncryptInit_ex` eingeleitet. Hierbei ist zu beachten, dass sowohl der verwendete Schlüssel von 256 Bit als auch der IV von 128 Bit die korrekten Längen aufweisen. Die Verschlüsselung erfolgt ohne zusätzliche Fülldaten (Padding) durch die Anweisung `EVP_CIPHER_CTX_set_padding`. Diese Konfiguration setzt jedoch voraus, dass die Eingabelänge ein Vielfaches der Blockgröße von 512 Byte beträgt.

```
void CryptHelper::InitDecrypt(){
    if(!(ctx = EVP_CIPHER_CTX_new())){
        errorMessage = HandleErrors();
    }
    if(1 != EVP_EncryptInit_ex(ctx, EVP_aes_256_cbc(), NULL, m_key, m_iv)){
        errorMessage = HandleErrors();
    }
    if(1 != EVP_CIPHER_CTX_set_padding(ctx, 0)){
        errorMessage = HandleErrors();
    }
}
```

Abbildung 4.11.: Initialisierung der AES-Verschlüsselung

Nach Abschluss der Initialisierungsphase erfolgt die Verschlüsselung der Datenblöcke. Die Datenblöcke werden hierzu, wie in Abbildung 4.12 dargestellt, sequenziell in die `UpdateEncrypt()`-Methode übergeben und im Rahmen dieses Prozesses die Funktion

EVP_EncryptUpdate für jeden Datenblock aufgerufen. Das Ziel dieser Funktion besteht in der blockweisen Verschlüsselung der Daten nach dem AES-CBC-Verfahren. Die resultierenden verschlüsselten Bilddaten werden in ein vorgesehenes Array gespeichert.

```
//Funktionsaufruf UpdateEncrypt()
unsigned char* encryptedBuffer = new unsigned char[BlockSize];
for(int position = 0; position < arrSize; position+=BlockSize) {
    helper.UpdateEncrypt(arr+position, BlockSize, encryptedBuffer);
    stream->WriteBuffer(encryptedBuffer, BlockSize);
}
//Funktion
int CryptHelper::UpdateEncrypt(unsigned char *input, int input_len, unsigned char *output){
    int processedBytes = 0;
    if(1 != EVP_EncryptUpdate(ctx, output, &processedBytes, input, input_len)){
        errorMessage = HandleErrors();
    }
    m_encrypt_len += processedBytes;
    return m_encrypt_len;
}
```

Abbildung 4.12.: Blockweise AES-Verschlüsselung der Bilddaten

Nach der Verarbeitung aller Bilddatenblöcke zeigt Abbildung 4.13 die Methode FinishEncrypt() mit EVP_EncryptFinal_ex, um zu veranschaulichen, wie der Verschlüsselungsprozess abgeschlossen wird. In dieser Phase besteht gegebenenfalls die Möglichkeit, zusätzliche verschlüsselte Bytes zu generieren, wodurch das Array vervollständigt wird. Der Kontext wird schließlich durch die Freigabe mittels der Funktion EVP_CIPHER_CTX_free freigegeben, um eine sichere Verwaltung des Speichers zu gewährleisten.

```
int CryptHelper::FinishDecrypt(unsigned char *output){
    int processedBytes = 0;
    if(1 != EVP_DecryptFinal_ex(ctx, output, &processedBytes)){
        errorMessage = HandleErrors();
    }
    m_decrypt_len += processedBytes;
    EVP_CIPHER_CTX_free(ctx);
    return m_decrypt_len;
}
```

Abbildung 4.13.: Abschluss der AES-Verschlüsselung

Der Entschlüsselungsprozess erfolgt in Analogie zur Verschlüsselung. Die zugrunde liegenden Methoden und Schritte sind ähnlich, wobei anstatt der Verschlüsselungsaufgaben die Entschlüsselungsoperationen `EVP_Decrypt` aufgerufen werden, um die ursprünglichen Daten rekonstruieren zu können. Nach erfolgreicher Verschlüsselung wird das Ergebnisbild in seiner verschlüsselten Form in Abbildung 4.14 dargestellt. Die Darstellung verdeutlicht, dass das Bild ohne den entsprechenden Schlüssel und Initialisierungsvektor nicht ohne Weiteres interpretiert oder entschlüsselt werden kann. Während in dieser prototypischen Umsetzung ein Landschaftsbild verwendet wird, gilt dasselbe Prinzip auch für die Verschlüsselung sensibler Patientenbilder in OnyxCeph^{3TM}. Die Herausforderung besteht darin, die verschiedenen Bildtypen zu berücksichtigen, die im realen Einsatz auftreten können. Die prototypische Umsetzung unterstreicht jedoch die Effektivität der AES-Verschlüsselung.



Abbildung 4.14.: AES-256-CBC Verschlüsseltes Landschaftsbild

4.2. Kapselung externer Bibliotheken

Externe Bibliotheken gewinnen in der Softwareentwicklung immer mehr an Bedeutung, stellen aber auch ein nicht zu unterschätzendes Sicherheitsproblem dar. Als Beispiel für Schwachstellen in externen Bibliotheken wurde im vorherigen Abschnitt bereits der Heartbleed-Bug in OpenSSL beschrieben. Ein weiteres Beispiel ist die Schwachstelle in der Bibliothek Log4j, die Ende 2021 für Aufsehen sorgte. Die Schwachstelle gefährdete Systeme, indem sie Angreifern potenziell die Remotecodeausführung ermöglichte. Nutzer der Bibliothek Log4j waren dadurch einem erhöhten Risiko ausgesetzt, das den Diebstahl oder die Manipulation sensibler Daten ermöglichte. [BSI, 2025g] Derartige Vorfälle verdeutlichen die Notwendigkeit, die Sicherheit im Umgang mit externen Bibliotheken zu erhöhen. Um dieser Herausforderung effektiv zu begegnen, wird der Ansatz der Kapselung externer Bibliotheken verfolgt. Durch die Isolierung solcher Bibliotheken kann der Hauptprozess vor potenziellen Sicherheitslücken geschützt werden. Für eine erfolgreiche Umsetzung des Kapselungsansatzes ist es jedoch zunächst notwendig, einen Überblick darüber zu erlangen, wie Prozesse unter Windows definiert werden.

4.2.1. Grundlagen von Prozessen unter Windows

Prozesse stellen in der Architektur des Windows-Betriebssystems die Ablaufumgebung für ein Programm zur Verfügung und sind somit eine dynamische Abfolge von Aktionen mit entsprechenden Zustandsänderungen. Ein Prozess umfasst dabei mehrere Attribute und Komponenten: [Mandl, 2014]

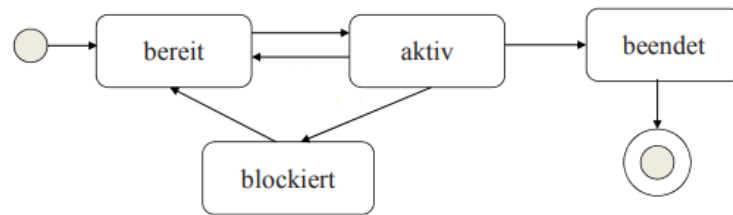
- Virtueller Adressraum: Ein isolierter Speicherbereich, der einem Prozess zugewiesen wird, um die Ausführung des Programms von anderen Prozessen zu trennen
- Prioritätsklasse: Bestimmt die Wichtigkeit des Prozesses im Vergleich zu anderen Prozessen, was sich z.B. auf die Zuteilung von CPU-Zeit auswirkt
- Geladene Programmmodule: Enthalten die ausführbaren Dateien und dynamische Bibliotheken (DLLs), die der Prozess benötigt
- Umgebungs- und Konfigurationsinformationen: Beinhaltet Parameter und Einstellungen, die den Ausführungskontext eines Prozesses definieren
- Mindestens ein Thread

Ein Prozess wird im Windows-System vom Kernel über den Objektmanager verwaltet und enthält alle notwendigen Zustandsinformationen, die in der Prozesstabelle gespeichert werden. Ein Element der Prozesstabelle ist der Prozesskontrollblock (PCB). Der PCB speichert alle für die Prozesssteuerung notwendigen Informationen: [Mandl, 2014]

- Programmzähler: Die Adresse des nächsten auszuführenden Befehls
- Prozessstatus: Der aktuelle Zustand des Prozesses
- Statische und dynamische Priorität: Gibt an, wie der Scheduler die Prozesse priorisiert.
- Verbrauchte Prozessorzeit: Dokumentiert die bisher verbrauchte CPU-Zeit seit dem Start des Prozesses
- Eindeutige Prozesskennung (PID) sowie die Prozesskennung des erzeugenden Prozesses (Elternprozess)
- Zugewiesene Ressourcen wie Dateien und Netzwerkverbindungen
- Aktuelle Registerinhalte: Die aktuellen Werte der CPU-Register des Prozesses

Die Verwaltung von Prozessen führt dazu, dass Prozesse während ihres Lebenszyklus verschiedene Zustände durchlaufen. Wie der Zustandsautomat in Abbildung 4.15 zeigt, werden Prozesse unter Windows typischerweise in die folgenden Zustände unterteilt: [Microsoft, 2025c]

- Bereit: Der Prozess wartet darauf, dass ihm Ressourcen zur Verfügung gestellt werden.
- Aktiv: Der Prozess wird gerade von der CPU ausgeführt.
- Blockiert: Der Prozess wartet auf die Verfügbarkeit einer benötigten Ressource.
- Beendet: Der Prozess hat seine Arbeit beendet und wird aus dem Speicher entfernt.



Quelle: [Mandl, 2014]

Abbildung 4.15.: Zustandsautomat eines Prozesses

Obwohl ein Prozess die Programmmodule, den Kontext und die Umgebung enthält, ist er nicht direkt für die Ausführung auf einem Prozessor vorgesehen. Stattdessen übernehmen die zu einem Prozess gehörenden Threads die tatsächliche Ausführung auf einem Prozessor. Da die Anzahl der Prozessoren in einem System begrenzt ist, können nicht alle Threads gleichzeitig ausgeführt werden. Aus diesem Grund implementiert Windows ein Konzept der zeitlichen Prozessorfreigabe. Dabei kann ein Thread für eine bestimmte Zeit ausgeführt werden, bevor der Prozessor zu einem anderen Thread wechselt. Dieser Wechselvorgang wird als Kontextwechsel bezeichnet und von der Windows-Komponente Dispatcher gesteuert. [Microsoft, 2025c]

Ein weiterer Aspekt bei der Verwendung von Prozessen ist, dass Prozesse standardmäßig nicht ohne weiteres miteinander kommunizieren können. Für die Kommunikation zwischen Prozessen sind spezielle Mechanismen wie die Interprozesskommunikation (IPC) notwendig. Darüber hinaus können Prozesse zu sogenannten Jobobjekten gruppiert werden, um gemeinsame Konfigurationen und Ressourcennutzung zu verwalten.

4.2.2. Implementierung der Kapselung externer Bibliotheken

Aufbauend auf den Grundlagen der Prozesse ist es das Ziel der folgenden Abschnitte, die Sicherheitsrisiken, die von externen Bibliotheken ausgehen können, zu minimieren. Dazu wird ein geeignetes Szenario definiert und prototypisch umgesetzt. In diesem Szenario werden Bilder im TIFF-Format verwendet, für deren Verarbeitung eine externe Bibliothek benötigt wird. Als Werkzeug zur Verarbeitung von TIFF-Bilddaten wird die Bibliothek LibTIFF eingesetzt. Sie ermöglicht das Einlesen und Konvertieren der Bilddaten in eine interpretierbare Form. [LibTIFF, 2025] Zur Realisierung dieses Szenarios werden zwei getrennte Prozesse entwickelt, der Hauptprozess und ein Nebenprozess. Der Hauptprozess steuert die Benutzeroberfläche und ist für die Darstellung der verarbeiteten Bilder

verantwortlich. Der Nebenprozess hingegen verwendet LibTIFF, um die Bilddaten zu verarbeiten. Dazu liest er die TIFF-Dateien ein und wandelt deren Daten in ein für den Hauptprozess verständliches Format um. Durch die klare Aufgabentrennung bleibt der Hauptprozess vor den Operationen der externen Bibliothek geschützt.

Einfache Kapselung der externen Bibliothek

Um die Trennung der externen Bibliothek zu implementieren, wird die Windows-API-Funktion `CreateProcessW` verwendet, um den Kindprozess zu initialisieren, welcher für die Ausführung der Bildverarbeitung verantwortlich ist. Um die notwendigen Parameter zu übergeben, wird ein String mit Argumenten vorbereitet. Der Codeabschnitt 4.16 zeigt, dass der String den Pfad zur ausführbaren Datei des Kindprozesses sowie zur TIFF-Bilddatei enthält. Hier werden Backslashes (`\`) verwendet, um die verschiedenen Teile des Strings für das Betriebssystem korrekt zu trennen. Dieser String wird dann konvertiert, um mit der Funktion `CreateProcessW` kompatibel zu sein.

```
String argString = "\"" + "Pfad zur Kindprozess.exe" +  
"\" + "Pfad zur TIFF-Bilddatei" + "\"";  
wchar_t *arg = argString.w_str();
```

Abbildung 4.16.: Vorbereitung des Argument-Strings für `CreateProcessW`

Mit den notwendigen Vorbereitungen kann die Funktion `CreateProcessW` wie in Abbildung 4.17 gezeigt verwendet werden, um den Kindprozess zu starten. Diese Funktion verwendet den vorbereiteten String mit einer Reihe von Parametern, deren Werte und Bedeutungen im Folgenden beschrieben werden: [Microsoft, 2025e]

- `lpApplicationName`: Dieser Parameter hat den Wert `NULL`, was zur Folge hat, dass der vollständige Pfad zur ausführbaren Datei in der `lpCommandLine` angegeben werden muss.
- `lpCommandLine`: Enthält den vollständigen Pfad und die Argumente für die Ausführung des Kindprozesses.
- `lpProcessAttributes` und `lpThreadAttributes`: Beide Parameter sind `NULL`, wodurch die Vererbung spezifischer Sicherheitseigenschaften an den neuen Prozess und Thread verhindert wird.

- `bInheritHandles`: Wenn dieser Parameter auf `FALSE` gesetzt ist, werden keine Handles vom aufrufenden Prozess an den neuen Prozess vererbt.
- `dwCreationFlags`: Auf 0 gesetzt bedeutet, dass keine speziellen Generierungs-Flags benötigt werden. Dies ermöglicht eine Prozessgenerierung ohne zusätzliche Prioritätsflags und Optionen.
- `lpEnvironment`: Hat den Wert `NULL`, was bedeutet, dass der neue Prozess die Umgebung des aufrufenden Prozesses verwendet.
- `lpCurrentDirectory`: Hat ebenfalls den Wert `NULL`, somit erbt der neue Prozess das aktuelle Verzeichnis des aufrufenden Prozesses.
- `lpStartupInfo`: Ein Zeiger auf eine `STARTUPINFO`-Struktur, die die Initialisierung des Prozesses enthält, einschließlich Fensterattributen und Handle-Einstellungen.
- `lpProcessInformation`: Eine Struktur, die Identifikationsinformationen über den neuen Prozess enthält, wie das Prozess-Handle und das Thread-Handle. Die Handles müssen mit der `CloseHandle`-Funktion geschlossen werden, um die Ressourcen freizugeben.

```
if (!CreateProcessW(
    NULL, // [in, optional] lpApplicationName
    arg,  // [in, out, optional] lpCommandLine
    NULL, // [in, optional] lpProcessAttributes
    NULL, // [in, optional] lpThreadAttributes
    FALSE, // [in] bInheritHandles
    0,     // [in] dwCreationFlags
    NULL, // [in, optional] lpEnvironment
    NULL, // [in, optional] lpCurrentDirectory
    &si,   // [in] lpStartupInfo
    &pi)   // [out] lpProcessInformation
) {
    // Fehlerbehandlung
}
```

Abbildung 4.17.: Aufruf der Funktion `CreateProcessW`

Im separaten Prozess wird nun das Argument `argv` verwendet, um den Datenpfad zur TIFF-Bilddatei zu ermitteln. Mit Hilfe von LibTIFF werden die Bilddaten verarbeitet und in einer Datei gespeichert, die dem in Abbildung 4.4 dargestellten Format ähnlich ist. Diese Datei wird im Dateisystem gespeichert und ist vom Hauptprozess aus zugänglich. Nach der Verarbeitung wird der Kindprozess beendet. Der Hauptprozess liest die vom Kindprozess erzeugte Bilddatei von der Festplatte und ermöglicht es dem Benutzer, das Bild anzuzeigen. Es ist jedoch zu beachten, dass die Bilddatei im Klartext auf dem System gespeichert wird und somit auch für unbefugte Benutzer zugänglich ist, was wiederum die Sicherheit beeinträchtigt. Aus diesem Grund wird im folgenden Abschnitt die Kombination von Verschlüsselung und Kapselung betrachtet, um die Vertraulichkeit der Daten zu gewährleisten.

Kombination von Verschlüsselung und Kapselung der externen Bibliothek

In diesem Abschnitt werden die bisherigen Überlegungen zur Kapselung und Verschlüsselung kombiniert. Der Ansatz besteht darin, die TIFF-Bibliothek zu kapseln und die Bilddaten mit der von OpenSSL bereitgestellten AES-Verschlüsselung im CBC-Modus zu verschlüsseln, bevor die Bilddatei auf dem System gespeichert wird. Für die Implementierung müssen sowohl der Hauptprozess als auch der Nebenprozess die OpenSSL-Bibliothek einbinden, um die Daten zu verschlüsseln und zu entschlüsseln. Als Einstiegspunkt wird der einfache Kapselungsansatz verwendet, welcher um die OpenSSL-Bibliothek, einen 256-Bit-Schlüssel und einen 128-Bit-Initialisierungsvektor erweitert wird. Der Schlüssel und der IV werden bei jedem Programmstart neu generiert. Beide werden dann an den Argumentstring von Abbildung 4.16 angehängt und über die `lpCommandLine` der Funktion `CreateProcessW` an den Kindprozess übergeben.

Im Kindprozess wird die TIFF-Bilddatei mit Hilfe der LibTIFF-Bibliothek eingelesen und die Bilddaten direkt aus dem TIFF-Format extrahiert. Der vereinbarte Schlüssel und der IV können dem `argv`-Array entnommen werden, so dass die Bilddaten vor der Speicherung auf dem System blockweise mit dem AES-256-CBC-Algorithmus verschlüsselt werden können. Sobald die Speicherung der Bilddaten über einen Stream abgeschlossen ist, wird der Kindprozess beendet. Auf diese Weise sind die Daten auf dem System vor unberechtigtem Zugriff geschützt.

Obwohl der kombinierte Ansatz die Bilddaten auf dem System durch Verschlüsselung schützt und somit einen unbefugten Zugriff verhindert, gibt es dennoch Sicherheitsaspek-

te, die berücksichtigt werden sollten. Insbesondere die Übergabe von Schlüssel und IV über die Kommandozeile birgt das Risiko eines Man-in-the-Middle-Angriffs, bei dem ein Angreifer die Übergabewerte abfangen und missbrauchen könnte. Darüber hinaus führt die Notwendigkeit, die OpenSSL-Bibliothek in beide Prozesse zu integrieren, zu einer weiteren Abhängigkeit von einer externen Bibliothek. Angesichts der oben genannten Risiken wird deutlich, dass die Kombination von Verschlüsselung und Kapselung von der Sicherheit des Schlüsselaustauschs abhängt. Deshalb wird im folgenden Abschnitt eine alternative Strategie betrachtet, deren Ziel es ist, keine Bilddatei auf dem System zu speichern, sondern einen Kommunikationsweg für die Bilddaten zwischen den Prozessen zu ermöglichen. Eine Verschlüsselung wäre dann nicht mehr notwendig.

Kapselung der externen Bibliothek mit Hilfe von Interprozesskommunikation

Gegenstand dieses Abschnitts ist die praktische Umsetzung der Interprozesskommunikation (IPC). Ziel ist die prototypische Implementierung der IPC unter Verwendung von Pipes zur Übertragung von Bilddaten zwischen einem Kindprozess und einem Hauptprozess. Dieses Szenario baut auf den im vorhergehenden Abschnitt 4.2.2 vorgestellten Überlegungen zur einfachen Kapselung der externen Bibliothek auf und erweitert diese um den Einsatz der IPC.

IPC ist ein Mechanismus, um die Kommunikation und den Datenaustausch zwischen Prozessen zu ermöglichen. Im Gegensatz zu Threads, die innerhalb eines einzelnen Prozesses arbeiten, erfordert die Kommunikation zwischen Prozessen bestimmte Kommunikationskanäle, die zur asynchronen Kommunikation verwendet werden. Diese Kanäle ermöglichen das Senden und Empfangen von Nachrichten über Prozessgrenzen hinweg. Zur Umsetzung stehen dazu eine Vielzahl von IPC-Mechanismen wie Message Queues, Pipes, Sockets oder FIFOs zur Verfügung, welche je nach Anwendungsfall eingesetzt werden können. [Wietzke and Tran, 2005] Für die prototypische Implementierung werden Pipes verwendet, um die Bilddaten vom Kindprozess zum Hauptprozess zu übertragen.

Pipes sind eine verbreitete Methode der Interprozesskommunikation, die eine Punkt-zu-Punkt-Verbindung zwischen zwei Prozessen darstellt. Somit dienen Pipes als Bereich eines gemeinsam genutzten Speichers, in den ein Prozess Daten schreiben kann, die ein anderer Prozess liest. Unterschieden wird zwischen anonymen und bekannten Pipes. Anonyme Pipes werden vorzugsweise für die Kommunikation zwischen verwandten Prozessen auf demselben Computer verwendet, insbesondere um die Standardeingabe und -ausgabe ei-

nes untergeordneten Prozesses umzuleiten. Bekannte Pipes hingegen ermöglichen die Datenübertragung zwischen nicht verwandten Prozessen und können auch über Netzwerke hinweg eingesetzt werden. [Microsoft, 2025j] Innerhalb dieses Szenarios werden anonyme Pipes verwendet.

Um die aus der TIFF-Datei erzeugten Bilddaten vom Kindprozess zum Hauptprozess zu übertragen, ist es notwendig, im Hauptprozess eine anonyme Pipe anzulegen. Diese Pipe dient als Kommunikationskanal und ermöglicht das Abfangen der Standardausgabe des Kindprozesses. Der Hauptprozess übernimmt die Verantwortung für die Initiierung und Verwaltung dieser Pipe-Verbindung. Zunächst wird im Hauptprozess mit der Funktion `CreatePipe` eine Pipe erzeugt. Dabei werden zwei Handles erzeugt, `childStdOUTRd` für das Lesen und `childStdOUTWr` für das Schreiben von Daten. Um eine sichere Kommunikation zu gewährleisten, wird ein Sicherheitsattribut konfiguriert. Dieses erlaubt es, die Vererbungseigenschaften der Handles zu definieren. Es ist wichtig, die Vererbungseigenschaften des Lesehandles zu bearbeiten, um sicherzustellen, dass dieser nicht vererbt wird. Die entsprechende Implementierung ist in Abbildung 4.18 dargestellt.

```
SECURITY_ATTRIBUTES securityAttr;
securityAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
securityAttr.bInheritHandle = TRUE;
securityAttr.lpSecurityDescriptor = NULL;
HANDLE childStdOUTRd = NULL; // Lese-Handle
HANDLE childStdOUTWr = NULL; // Schreib-Handle

if (!CreatePipe(&ghChildStdOUTRd,
               &ghChildStdOUTWr,
               &securityAttr,
               0 // Standardpuffergröße
               )) {
    // Fehlerbehandlung Pipe wurde nicht erstellt
}
if (!SetHandleInformation(ghChildStdOUTRd, HANDLE_FLAG_INHERIT, 0)) {
    // Fehlerbehandlung Read-Pipe wird vererbt
}
```

Abbildung 4.18.: Erstellung einer anonymen Pipe

Nach erfolgreicher Erstellung der Pipe wird der Kindprozess mit der Funktion `CreateProcess` gestartet und seine Standardausgabe in die Pipe umgeleitet. Der Parameter `bInheritHandles` der Funktion wird auf `TRUE` gesetzt und stellt sicher, dass das benötigte Handle vererbt wird. Die Implementierung ist im Codeausschnitt 4.19 dargestellt.

```

PROCESS_INFORMATION pi;
STARTUPINFO si;
ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
ZeroMemory(&si, sizeof(STARTUPINFO));
si.cb = sizeof(STARTUPINFO);
si.hStdError = childStdOUTWr;
si.hStdOutput = childStdOUTWr;
si.dwFlags |= STARTF_USESTDHANDLES;

if(!CreateProcess(NULL, arg, NULL, NULL, TRUE, 0, NULL, NULL, &si, &pi))
{
    // Fehlerbehandlung Prozess nicht gestartet!
}

```

Abbildung 4.19.: Starten des Kindprozesses mit CreateProcess

Im Kindprozess erfolgt schließlich die Übertragung der Bilddaten, nachdem die TIFF-Daten mit Hilfe der Bibliothek eingelesen und in ein interpretierbares Format umgewandelt wurden. Dabei werden die Daten sequenziell in die zuvor erstellte Pipe geschrieben. Diese sequenzielle Übertragung ist wichtig, da Pipes sicherstellen, dass die Daten in der Reihenfolge empfangen werden, in der sie gesendet wurden. Der Prozess der Übertragung von Bilddaten ist in Abbildung 4.20 veranschaulicht.

```

THandleStream *stream = new THandleStream(STDOUTHANDLE);
try {
    stream->WriteBuffer((unsigned char*)&width, 4);
    stream->WriteBuffer((unsigned char*)&height, 4);
    stream->WriteBuffer((unsigned char*)&channel, 4);
    stream->WriteBuffer((unsigned char*)&bitPerChannel, 4);

    for (int position = 0; position < pixelDataSize; position += BlockSize) {
        stream->WriteBuffer(pixelData + position, BlockSize);
    }
} catch (Exception &E) {
    // Fehlerbehandlung
} catch (...) {
    // Fehlerbehandlung
}
delete stream;

```

Abbildung 4.20.: Sequenzielles Schreiben von Bilddaten in die Pipe

Nach erfolgreicher Übertragung der Bilddaten über die Pipe kann der Hauptprozess die Daten empfangen und auf der Benutzeroberfläche darstellen. Dazu liest er die Daten aus dem Handle `childStdOUTRd` und bereitet sie für die Darstellung auf der Benutzeroberfläche auf. Nach Abschluss der Übertragung wird der Kindprozess beendet. Es ist wichtig, sowohl das Lesehandle `childStdOUTRd` als auch das Schreibhandle `childStdOUTWr` und die Prozesshandles nach der Verarbeitung der Daten zu schließen, um Ressourcen freizugeben und mögliche Speicherlecks zu vermeiden.

Die Verwendung von Pipes für die Interprozesskommunikation ermöglicht eine sichere Übertragung von Bilddaten zwischen Kindprozess und Hauptprozess, ohne dass unverschlüsselte Daten auf dem System gespeichert werden. Dieser Ansatz behebt die Sicherheitsrisiken der bisherigen Ansätze und sorgt für eine klare Trennung der TIFF-Bibliothek. Gleichzeitig bietet der Ansatz eine stabile Basis für mögliche Erweiterungen, wie die Einführung von Sandboxing-Techniken zur zusätzlichen Absicherung.

4.2.3. Implementierung von Sandboxing-Techniken

Um die Ausführung potenziell schädlicher Anwendungen in einer kontrollierten Umgebung zu ermöglichen und damit das Gesamtsystem vor nicht autorisierten Änderungen zu schützen, können Sandbox-Techniken in der Cybersicherheit eingesetzt werden. Die hier vorgestellte Implementierung baut auf dem bestehenden Konzept der Interprozesskommunikation auf und erweitert es um zusätzliche Sicherheitsmaßnahmen. Die Implementierung basiert auf der Sandbox-Architektur von Chromium, welche eine geschützte Ausführungsumgebung unter Verwendung bewährter Methoden zur Prozess- und Ressourcenisolierung bereitstellt. Die Besonderheit der Chromium Sandbox-Architektur liegt in der Verwendung von systemintegrierten Sicherheitsmechanismen von Microsoft. Dazu gehören Auftragsobjekte (Job-Objekte), benutzerdefinierte Zugriffskontrolllisten (DACLS), Integritätsebenen und eingeschränkte Tokens. Diese Komponenten arbeiten zusammen, um eine Umgebung zu schaffen, in der Prozesse mit eingeschränkten Zugriffsrechten ausgeführt werden und die potenzielle Angriffsfläche reduziert wird. [Google, 2025]

Im Folgenden wird dargelegt, wie die zusätzlichen Sicherheitsmechanismen in der Praxis implementiert werden. Um die Integration der einzelnen Komponenten klar und verständlich darzustellen, wird jeder Schritt erläutert und durch entsprechende Codebeispiele verdeutlicht. Dabei werden die Fehlerfälle bewusst ausgelassen, um den Fokus auf die Funktionalität und das Zusammenspiel der Komponenten zu legen, ohne den Rahmen zu sprengen.

Zu Beginn werden die Zugriffsrechte eines Prozesses mithilfe eines eingeschränkten Tokens reduziert. Das eingeschränkte Token ist dabei die Sicherheitsmaßnahme, welche die Berechtigungen des Prozesses einschränkt. Es wird aus dem Token des Hauptprozesses erzeugt und mithilfe der in Abbildung 4.21 aufgeführten Methode `CreateRestrictedTokenFromProcess()` erstellt. Innerhalb dieser Methode wird die Funktion `CreateRestrictedToken` genutzt, um Berechtigungen zu entfernen und bestimmte Sicherheitskennungen (SIDs) einzuschränken. [Microsoft, 2025h] Letzteres wurde in diesem Fall jedoch nicht umgesetzt. Stattdessen wird das Flag `DISABLE_MAX_PRIVILEGE` gesetzt, wodurch alle nicht notwendigen Rechte deaktiviert werden. [Microsoft, 2025f]

```
HANDLE SandboxHelper::CreateRestrictedTokenFromProcess() {
    HANDLE hToken = NULL;
    if (!OpenProcessToken(GetCurrentProcess(), TOKEN_ALL_ACCESS, &hToken)) {
        //ERROR: OpenProcessToken failed
    }
    HANDLE restrictedToken = NULL;
    if (!CreateRestrictedToken(hToken, DISABLE_MAX_PRIVILEGE,
        0, NULL, 0, NULL, 0, NULL, &restrictedToken)) {
        //ERROR: CreateRestrictedToken failed
    }
    CloseHandle(hToken);
    return restrictedToken;
}
```

Abbildung 4.21.: Erstellung eines eingeschränkten Tokens

Nachdem das Token erstellt wurde, wird dessen Integritätsstufe auf den Wert Niedrig gesetzt. Integritätsstufen in Windows dienen als Form der Zugriffskontrolle, indem sie die Interaktionsmöglichkeiten eines Prozesses mit sensibleren Systemressourcen regeln. Hierzu wird die Methode `SetLowIntegrityLevel()` verwendet, deren Umsetzung in Abbildung 4.22 veranschaulicht wird. Die zugehörige SID wird mit der Funktion `AllocateAndInitializeSid` erstellt und dient als eindeutiges Identifikationsmerkmal, welches den Sicherheitskontext des Prozesses angibt. Dadurch wird sichergestellt, dass ein Prozess ausschließlich auf Ressourcen zugreifen kann, die denselben oder einen niedrigeren Integritätsschutz aufweisen. [Microsoft, 2025] Zu den erlaubten Zugriffen gehören unter anderem: [Google, 2025]

- Lesender Zugriff auf die meisten Dateien
- Schreibzugriff auf `%USERPROFILE%\AppData\LocalLow`
- Lesender Zugriff auf die meisten Einträge der Windows-Registry
- Schreibzugriff auf `HKEY_CURRENT_USER\Software\AppDataLow`

```
void SandboxHelper::SetLowIntegrityLevel(HANDLE token) {
    PSID lowIntegritySid = NULL;
    SID_IDENTIFIER_AUTHORITY integrityAuthority = SECURITY_MANDATORY_LABEL_AUTHORITY;
    if (!AllocateAndInitializeSid(&integrityAuthority, 1, SECURITY_MANDATORY_LOW_RID,
        0, 0, 0, 0, 0, 0, 0, &lowIntegritySid)) {
        //ERROR: AllocateAndInitializeSid failed
    }
    TOKEN_MANDATORY_LABEL TML;
    TML.Label.Attributes = SE_GROUP_INTEGRITY;
    TML.Label.Sid = lowIntegritySid;
    DWORD labelSize = sizeof(TOKEN_MANDATORY_LABEL) + GetLengthSid(lowIntegritySid);
    if (!SetTokenInformation(token, TokenIntegrityLevel, &TML, labelSize)) {
        //ERROR: SetTokenInformation failed
    }
    FreeSid(lowIntegritySid);
}
```

Abbildung 4.22.: Festlegung der Integritätsstufe auf Niedrig

Im weiteren Verlauf wird ein Auftragsobjekt eingerichtet, das die Verwaltung und Kontrolle von Prozessen als Einheit ermöglicht. [Microsoft, 2025a] Die Methode `CreateJobObjectWithDACL()` erstellt das Auftragsobjekt und weist ihm eine DACL zu. Über die DACL wird festgelegt, welche Benutzer oder Prozesse Zugriff auf das Auftragsobjekt erhalten. [Microsoft, 2025g] Dieser Vorgang ist in Abbildung 4.23 dargestellt. Um sicherzustellen, dass alle Benutzer Zugriff auf das Job-Objekt haben, wird für die Benutzergruppe eine SID erstellt. Der Zugriff wird in einem Zugriffseintrag der DACL definiert, wodurch den Benutzern Rechte auf das Job-Objekt gewährt werden.

```
HANDLE SandboxHelper::CreateJobObjectWithDACL() {
    HANDLE hJob = CreateJobObject(NULL, NULL);
    if (hJob == NULL) {
        //ERROR: CreateJobObject failed
    }
    PSID pSID = NULL;
    SID_IDENTIFIER_AUTHORITY sidAuthority = SECURITY_WORLD_SID_AUTHORITY;
    if (!AllocateAndInitializeSid(&sidAuthority, 1, SECURITY_WORLD_RID,
        0, 0, 0, 0, 0, 0, 0, &pSID)) {
        //ERROR: AllocateAndInitializeSid failed;
    }
    EXPLICIT_ACCESS ea;
    ZeroMemory(&ea, sizeof(EXPLICIT_ACCESS));
    ea.grfAccessPermissions = JOB_OBJECT_ALL_ACCESS;
    ea.grfAccessMode = SET_ACCESS;
    ea.Trustee.TrusteeForm = TRUSTEE_IS_SID;
    ea.Trustee.TrusteeType = TRUSTEE_IS_GROUP;
    ea.Trustee.ptstrName = (LPTSTR)pSID;
    PACL pACL = NULL;
    DWORD result = SetEntriesInAcl(1, &ea, NULL, &pACL);
    if (result != ERROR_SUCCESS) {
        //ERROR: SetEntriesInAcl failed
    }
    result = SetSecurityInfo(hJob, SE_KERNEL_OBJECT, DACL_SECURITY_INFORMATION, NULL, NULL,
        pACL, NULL);
    if (result != ERROR_SUCCESS) {
        //ERROR: SetSecurityInfo failed
    }
    FreeSid(pSID);
    LocalFree(pACL);
    return hJob;
}
```

Abbildung 4.23.: Erstellung eines Auftragsobjekts mit DACL

Nach der Erstellung des Job-Objekts werden mithilfe der Methode `SetJobObjectLimits()` spezifische Limitierungen innerhalb dieses Auftragsobjekts gesetzt. Mithilfe dieser Methode werden Einschränkungen definiert, die der kontrollierten Nutzung und Verwaltung von Ressourcen dienen. Dazu gehören Beschränkungen der Benutzeroberfläche sowie die Begrenzung von Prozessen, die vor unerwünschten Manipulationen schützen. Die Liste der Einschränkungen kann dabei der Abbildung 4.24 entnommen werden.

```
HANDLE SandboxHelper::SetJobObjectLimits(HANDLE m_hJob) {
    JOBOBJECT_BASIC_LIMIT_INFORMATION jobLimits = {};
    JOBOBJECT_EXTENDED_LIMIT_INFORMATION extendedLimits = {};
    JOBOBJECT_BASIC_UI_RESTRICTIONS uiLimits = {};

    jobLimits.LimitFlags = JOB_OBJECT_LIMIT_KILL_ON_JOB_CLOSE |
        JOB_OBJECT_LIMIT_ACTIVE_PROCESS | JOB_OBJECT_LIMIT_DIE_ON_UNHANDLED_EXCEPTION;
    jobLimits.ActiveProcessLimit = 1;
    extendedLimits.BasicLimitInformation = jobLimits;

    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_WRITECLIPBOARD;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_READCLIPBOARD;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_HANDLES;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_GLOBALATOMS;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_DISPLAYSETTINGS;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_SYSTEMPARAMETERS;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_DESKTOP;
    uiLimits.UIRestrictionsClass |= JOB_OBJECT_UILIMIT_EXITWINDOWS;

    if (!SetInformationJobObject(m_hJob, JobObjectBasicUIRestrictions, &uiLimits, sizeof
        (uiLimits))) {
        //ERROR: Fehler bei UI-Beschränkungen
    }
    if (!SetInformationJobObject(m_hJob, JobObjectExtendedLimitInformation, &extendedLimits,
        sizeof(extendedLimits))) {
        //ERROR: Fehler bei erweiterten Beschränkungen
    }
    return m_hJob;
}
```

Abbildung 4.24.: Festlegung der Grenzen für das Auftragsobjekt

Nachdem alle Sicherheitsmaßnahmen implementiert wurden, ist der nächste Schritt, den Prozess mithilfe der Funktion `CreateProcessAsUserW` zu starten. Diese Funktion ähnelt weitgehend der Funktionsweise von `CreateProcess`, verwendet jedoch zusätzlich das eingeschränkte Token, um den Sicherheitskontext des neuen Prozesses festzulegen.

[Microsoft, 2025d] Sobald der Prozess erfolgreich gestartet wurde, wird er dem erstellten Auftragsobjekt zugewiesen. Dadurch werden sämtliche zuvor definierten Einschränkungen und Sicherheitsmaßnahmen auf den Prozess angewendet. Auf diese Weise wird gewährleistet, dass der Kindprozess in einer isolierten Umgebung ausgeführt wird.

Um die Funktionsweise der Implementierung zu verifizieren, werden spezifische Tests definiert und durchgeführt. Diese sind darauf ausgelegt, die Einschränkungen zu prüfen und sicherzustellen, dass unautorisierte Aktionen blockiert werden. Zu den durchgeführten Tests gehören:

- Test 1 und 2: Zugriff auf das Dateisystem. In diesem Test wird überprüft, ob der Prozess in der Lage ist, Dateien in unberechtigten Verzeichnissen zu erstellen oder zu verändern.
- Test 3: Erstellung von Desktop-Objekten. Dabei wird überprüft, ob unautorisierte Versuche, neue Desktops zu erzeugen, blockiert werden.
- Test 4: Zugriff auf Internetverbindungen. Hierbei wird geprüft, ob unautorisierte Internetverbindungen unterbunden werden.
- Test 5 und 6: Zugriff auf die Zwischenablage. Dabei wird überprüft, ob unberechtigte Lese- und Schreiboperationen in die Zwischenablage möglich sind.
- Test 7 und 8: Verwendung von globalen Atomen. Es wird überprüft, ob der Prozess auf das globale Atom des Vaterprozesses zugreifen kann.
- Test 9: Versuch der Systemabmeldung. Dabei wird ermittelt, ob der Prozess den Systemabmeldevorgang initiieren kann.
- Test 10 und 11: Änderung der Anzeige- und Systemeinstellungen. Dabei wird überprüft, ob unautorisierte Änderungen an den Systemeinstellungen und Anzeigekonfigurationen möglich sind.

Die in Abbildung 4.25 dargestellten Testergebnisse zeigen an, ob der jeweilige Test erfolgreich war oder eine Fehlermeldung angezeigt wurde. Dabei sind die folgenden Fehlercodes aufgetreten: [Microsoft, 2025n] [Microsoft, 2025o]

- ERRORACCESSDENIED (Code 5): Zugriff verweigert.
- ERRORFILENOT_FOUND (Code 2): Die angegebene Datei konnte nicht gefunden werden.
- ERRORNOTENOUGH_QUOTA (Code 1816): Die bereitgestellten Informationen reichen für die Verarbeitung dieses Befehls nicht aus.

Der daraus resultierende Output bestätigt, dass die Sandbox-Implementierung den Prozess einschränkt und somit die Sicherheit im System erhöht.

```
Test1 Fehler: Datai auf Laufwerk D erstellen - Code: 5
Test2 Fehler: Datei auf Laufwerk E erstellen - Code: 5
Test3 Fehler: Neuen Desktop erstellen - Code: 5
Test4 Fehler: Eine URL öffnen - Code: 1816
Test5 Fehler: SetClipboardData - Code: 5
Test6 Fehler: GetClipboardData - Code: 5
Test7: GlobalAddAtom innerhalb des JobObjekts erzeugt
Test8 Fehler: GlobalFindAtom - Code: 2
Test9 Fehler: ExitWindows - Code: 2
Test10 Fehler: DISPLAYSETTINGS - Code: 5
Test11 Fehler: SYSTEMPARAMETERS - Code: 5
```

Abbildung 4.25.: Ergebnisse der Sicherheits- und Funktionstests der Sandbox

4.3. Authentifizierung

Die Authentifizierung ist eine Sicherheitsmaßnahme, die sicherstellt, dass nur autorisierte Nutzer auf Systeme und Ressourcen zugreifen können. Ihr Ziel ist die Verifizierung von Identitäten und somit die Verhinderung unberechtigten Zugriffs. In diesem Abschnitt wird dargestellt, wie die Authentifizierung umgesetzt werden könnte, um die Software OnyxCeph^{3TM} widerstandsfähiger gegenüber Phishing-Angriffen zu machen. Dazu werden zunächst die notwendigen Grundlagen beschrieben.

4.3.1. Grundlagen der Authentifizierung

Bevor ein bestimmter Authentifizierungsansatz konzeptionell dargestellt werden kann, ist es wichtig, die verfügbaren Methoden zu verstehen. Im Folgenden werden deshalb die unterschiedlichen Faktoren Wissen, Besitz und Biometrie sowie die verschiedenen Arten von Authentifizierungsmethoden beleuchtet.

Bei der Authentifizierung kommen verschiedene Faktoren zum Einsatz, die zur Identifikation einer Person oder eines Geräts genutzt werden. Diese Faktoren lassen sich in folgende Kategorien einteilen: [Papathanasaki et al., 2022]

- Wissensbasierte Faktoren: Dazu zählen Informationen, zu denen nur der Benutzer Zugang hat. Das können beispielsweise Passwörter oder Sicherheitsfragen sein. Allerdings sind diese Faktoren anfällig für Angriffe wie Phishing und Brute-Force-Attacken, bei denen versucht wird, sich Zugang zu verschaffen, indem die jeweilige Information systematisch erraten wird.
- Besitzbasierte Faktoren: Der Nutzer muss ein physisches Objekt besitzen, das zur Verifizierung dient. Hardware-Tokens, wie spezielle USB-Sticks, sind sichere physische Verifizierungsmittel, da sie nicht ohne Weiteres entwendet werden können. Auch Software-Tokens, wie sie in Authentifizierungs-Apps auf Smartphones verwendet werden, benötigen das Gerät, um Einmalpasswörter zu generieren.
- Biometrische Faktoren: Dabei werden physische Merkmale genutzt, um den Benutzer zu verifizieren. Fingerabdruckscanner und Gesichtserkennung gewähren basierend auf einzigartigen biologischen Merkmalen Zugriff und sind dabei besonders schwer zu fälschen.

Durch den Einsatz verschiedener Faktoren ist die Umsetzung unterschiedlicher Authentifizierungsansätze möglich. Anschließend werden die einzelnen Ansätze genauer dargestellt: [Papathanasakis et al., 2022]

- Die Single-Faktor-Authentifizierung nutzt nur einen Authentifizierungsfaktor, typischerweise in Form eines Passworts. Diese Methode ist zwar einfach zu implementieren, bietet jedoch nur geringen Schutz. Das Problem dabei ist, dass ein Angreifer nur das Passwort benötigt, um Zugriff auf die Ressource zu erhalten.
- Die Zwei-Faktor-Authentifizierung kombiniert zwei unterschiedliche Authentifizierungsfaktoren. Ein alltägliches Beispiel findet sich bei der Anmeldung im Online-Banking, bei der Benutzer sowohl ein Passwort als auch ein per SMS gesendetes Einmalpasswort eingeben, mit dem sie sich authentifizieren. Diese zusätzliche Sicherheitsebene erschwert es Angreifern erheblich, Zugang zu erlangen.
- Die Multi-Faktor-Authentifizierung erweitert diese Ansätze durch die Integration zusätzlicher Faktoren, was die Sicherheit weiter erhöht. Ein Beispiel hierfür ist die Verwendung eines Passworts, einer Sicherheitsfrage und eines biometrischen Merkmals, wie etwa eines Fingerabdrucks, um höchsten Schutz zu gewährleisten.

Ein weiterer Aspekt bei der Umsetzung von Authentifizierungsmethoden ist die Definition der Kommunikationswege zwischen den Interaktionspartnern. Hierbei übernehmen Protokolle eine zentrale Rolle, da sie einen einheitlichen und standardisierten Ansatz bieten, um die Identität eines Nutzers zu bestätigen und ihm den Zugriff auf Ressourcen zu gewähren. Ein solches Protokoll ist OAuth 2.0, das als Industriestandard für die Autorisierung gilt. [Parecki, 2025] OAuth 2.0 definiert klar strukturierte Abläufe für den Zugriff auf Webanwendungen, Desktop-Apps, mobile Geräte und andere Plattformen. Auf diese Weise kann es einheitlich auf verschiedenen Plattformen realisiert werden.

4.3.2. Konzept zur Umsetzung einer Authentifizierungsmethode

Die Softwarelösung von OnyxCeph^{3TM} basiert derzeit auf einer Single-Faktor-Authentifizierung, bei der Benutzername und Passwort kombiniert werden. Um die Sicherheit weiter zu erhöhen, wird die Einführung einer Zwei-Faktor-Authentifizierung in Betracht gezogen. Neben dem bereits vorhandenen wissensbasierten Faktor soll ein besitzbasierter Faktor in Form eines Hardware-Tokens hinzugefügt werden. Dabei kommt der FIDO2-Standard zum Einsatz. [FIDO, 2025]

FIDO2 ist ein Benutzerauthentifizierungsstandard, der aus zwei von der FIDO-Allianz gemeinsam mit dem World Wide Web Consortium (W3C) entwickelten Komponenten besteht:

- Web Authentication API (WebAuthn): Eine Schnittstelle, welche die Erstellung schlüsselbasierter Anmeldeinformationen ermöglicht und ohne Passwörter überprüft werden kann. Der Anmeldevorgang basiert dabei auf einem Aufforderung-Antwort-Verfahren (engl. Challenge-Response). [W3C, 2025]
- Client-to-Authenticator Protocol (CTAP): Ein Protokoll, das die Kommunikation mit einem Authentifikator beschreibt. Beispielsweise zwischen einem USB-Token und dem Client, in der Regel einem Webbrowser. Es unterstützt Verbindungen über verschiedene Technologien wie USB, NFC und Bluetooth. [FIDO, 2019]

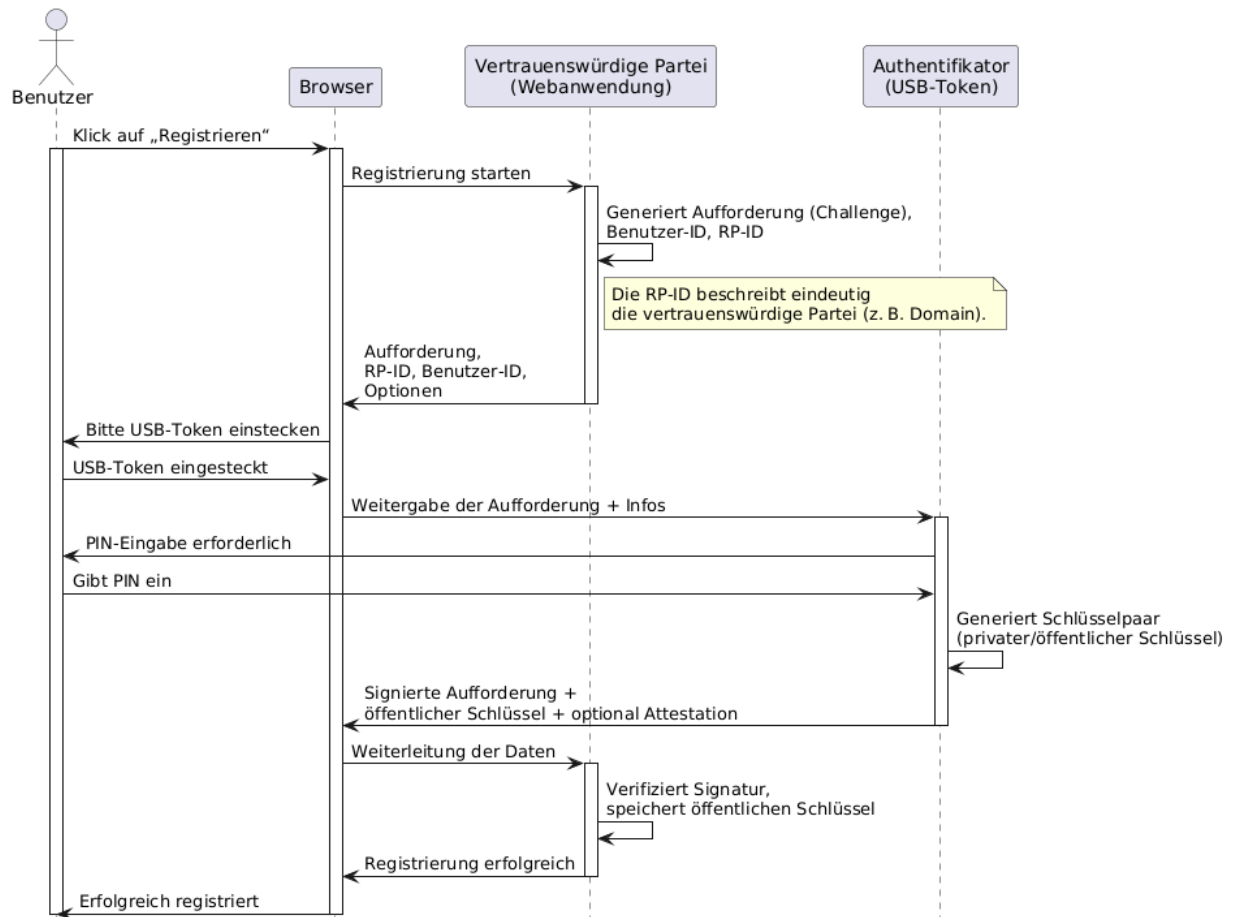
Beide Komponenten werden von der FIDO2-Authentifizierung genutzt, um durch asymmetrische Kryptografie die folgenden beiden Vorgänge umzusetzen:

- Registrierung: Der Authentifikator erzeugt ein Schlüsselpaar, wobei der private Schlüssel sicher auf dem Gerät verbleibt und der öffentliche Schlüssel an die vertrauenswürdige Partei (den Server) gesendet wird. Der Nachweis der Authentizität erfolgt typischerweise durch Interaktionen wie einen Fingerscan oder das Einstecken eines USB-Tokens.
- Authentifizierung: Bei der Anmeldung sendet der Server eine Aufforderung an den Client. Das Authentifizierungsgerät signiert diese Aufforderung mit dem privaten Schlüssel und sendet sie an den Server zurück, um die Identität des Nutzers sicher zu bestätigen.

Der Ablauf dieser Protokolle umfasst mehrere Schritte und reicht von der Erzeugung einer Aufforderung bis zur erfolgreichen Verifizierung auf dem Server. Die Registrierung und Authentifizierung werden im weiteren Verlauf anhand von Sequenzdiagrammen genauer erläutert. Für die Umsetzung von FIDO2 stehen Entwicklern verschiedene Frameworks zur Verfügung, darunter die WebAuthn-Rust-Bibliothek und das Yubico-SDK. Diese bieten flexibel anpassbare und sichere Lösungen. [Yubico, 2025][Kanidm, 2025] Unternehmen wie Yubico, Feitian und Kensington unterstützen FIDO2 und bieten robuste Token, die sich in bestehende Sicherheitsinfrastrukturen integrieren lassen. [Microsoft, 2025k]

Zur Veranschaulichung der praktischen Umsetzung einer FIDO2-Authentifizierung wird die von der FIDO Alliance bereitgestellte Demonstration genutzt. [FIDO, 2025] Mithilfe der Sequenzdiagramme aus Abbildung 4.26 für die Registrierung und Abbildung 4.27 für die Authentifizierung soll das FIDO Alliance Projekt als Referenz dienen. Das Ziel besteht darin, die notwendigen Schritte zur Implementierung von FIDO2 aufzuzeigen.

Während der Registrierung wird für jeden Nutzer ein öffentlich-privates Schlüsselpaar generiert und registriert. Hierbei durchläuft der Prozess die folgenden Schritte:



Quelle: [FIDO, 2025][Descope, 2025]

Abbildung 4.26.: Sequenzdiagramm zur FIDO2-Registrierung mit einem USB-Token

1. Erstellung von Zugangsdaten:

- Der Registrierungsvorgang beginnt mit der Erstellung eines Antrags.
- Der Aufforderungs-Parameter generiert einen 32 Byte langen, zufälligen Datenbuffer.
- Die vertrauenswürdige Partei (Server oder Webanwendung) muss detaillierte Informationen angeben, darunter den Organisationsnamen und eine eindeutige ID zur Identifikation von Authentifikationsanfragen.
- Es sind Benutzerinformationen erforderlich, einschließlich einer eindeutigen, zufällig generierten Benutzer-ID und eines Anzeigenamens. Dies stellt die Identität des Nutzers im System sicher.

2. Kommunikation mit dem Authentifikator:

- Die vom Server generierte Aufforderung wird an den Authentifikator (biometrischer Scanner, Sicherheitsschlüssel oder Smartphone) weitergeleitet.
- Der Authentifikator fordert eine Autorisierungsaktion (z. B. Fingerabdruckscan, Gesichtserkennung oder PIN-Eingabe) vom Nutzer.

3. Serverunterstützte Algorithmen:

- Die Implementierung muss die Unterstützung der Algorithmen RS1, RS256 und ES256 zur Verschlüsselung gewährleisten. Mithilfe dieser Algorithmen wird sichergestellt, dass die übertragenen Daten nicht verändert wurden.

4. Weitere Konfigurationen:

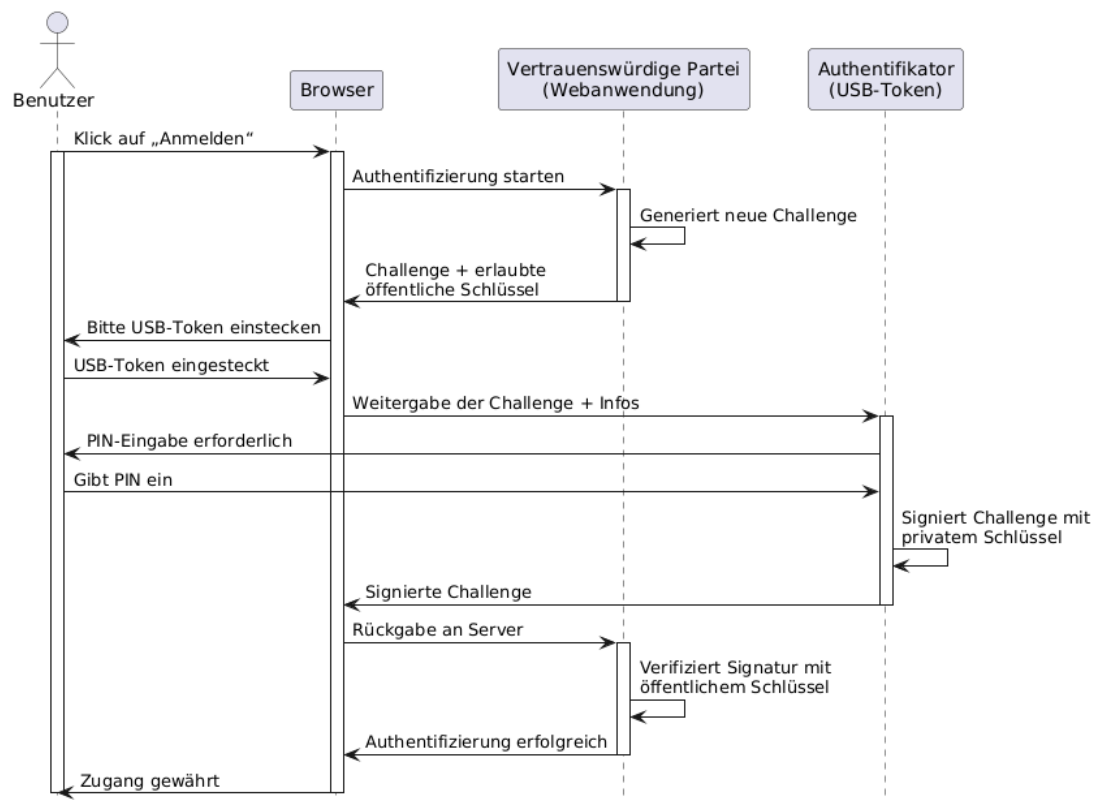
- Optional können Sicherheitsrichtlinien wie Zeitüberschreitungen und Anforderungen an Authentifikator-Typen hinzugefügt werden.
- Ebenso optional kann die Verarbeitung von Attestation-Daten ergänzt werden, um die Vertrauenswürdigkeit einer Plattform und die Integrität der darin ausgeführten Binärdateien remote zu überprüfen.

5. Abschluss der Registrierung:

- Der Authentifikator erstellt ein neues öffentlich-privates Schlüsselpaar, signiert die Aufforderung und sendet die signierten Daten sowie den öffentlichen Schlüssel an die vertrauenswürdige Partei.

- Die vertrauenswürdige Partei speichert den öffentlichen Schlüssel in Verbindung mit dem Nutzer, wodurch die Registrierung abgeschlossen wird.

Der Anmeldeprozess ist erforderlich, damit der Nutzer Zugang zur Anwendung erhält. Zur Umsetzung dieses Prozesses wird wie folgt vorgegangen:



Quelle: [FIDO, 2025][Descope, 2025]

Abbildung 4.27.: Sequenzdiagramm zur FIDO2-Authentifizierung mit einem USB-Token

1. Überprüfung der Challenge:

- Die vertrauenswürdige Partei generiert eine neue Aufforderung, ähnlich wie bei der Registrierung. Diese muss vom Benutzer mit seinem Authentifikator signiert werden.
- Die signierte Aufforderung wird an die vertrauenswürdige Partei, beispielsweise einen Server, zurückgesandt, welcher die Authentizität des Nutzers überprüft.

2. Zulässige Zugangsdaten prüfen:

- In diesem Schritt wird überprüft, welche zuvor registrierten Authentifizierungstokens für die Anmeldung zugelassen sind. Dadurch wird sichergestellt, dass nur autorisierte und vertrauenswürdige Geräte verwendet werden können.

3. Kommunikation mit dem Authentifikator:

- Die Informationen zum zuvor erstellten privaten Schlüssel werden an den entsprechenden Authentifikator weitergeleitet.
- Der Nutzer führt die Autorisierungsaktion aus, um die Berechtigung zum Weiterführen des Authentifizierungsvorgangs zu erhalten. Dies kann in Form eines physischen oder biometrischen Faktors erfolgen, beispielsweise durch das Berühren eines Fingerabdrucksensors, die Gesichtserkennung oder das Einstecken eines USB-Sicherheitsschlüssels.

4. Benutzerverifikation:

- In diesem Schritt wird die Identität des Nutzers durch den Authentifikator überprüft, um sicherzustellen, dass es sich um einen autorisierten Nutzer handelt.
- Die Identität wird mithilfe von Funktionen wie einem Fingerabdruckscanner oder der PIN-Eingabe eines zugehörigen Hardware-Tokens bestätigt. Nach erfolgreicher Verifizierung kann der Authentifikator den privaten Schlüssel verwenden, um die Aufforderung zu signieren und an die vertrauenswürdige Partei zurückzusenden.

5. Prüfung und Zugang:

- Die vertrauenswürdige Partei verifiziert die Signatur mithilfe des gespeicherten öffentlichen Schlüssels.
- Bei erfolgreicher Verifikation wird der Nutzer authentifiziert und erhält Zugang zur Anwendung.

Durch die Umsetzung der beschriebenen FIDO2-Mechanismen, die eine Registrierung und Authentifizierung mittels eines Hardware-Tokens ermöglichen, kann eine Zwei-Faktor-Authentifizierung realisiert werden. Dadurch wird die Sicherheit gegenüber Phishing- oder Man-in-the-Middle-Angriffen erhöht, da der Zugriff für Angreifer erschwert wird. Jedoch

steht der erhöhten Sicherheit ein entscheidender Nachteil gegenüber, nämlich die Notwendigkeit einer Internetverbindung. Die aktuelle Spezifikation von OnyxCeph^{3TM} enthält keine Vorgabe für eine Internetverbindung. Daraus ergibt sich ein Widerspruch zur Einführung dieses Sicherheitsstandards. Sollte die Spezifikation allerdings in Zukunft eine Internetverbindung vorsehen, wäre die Implementierung dieser Authentifizierungsmethode eine relevante Sicherheitsmaßnahme.

5. Auswertung

Nachdem die ausgewählten Cybersicherheitsmaßnahmen dargestellt wurden, ist eine Evaluation sinnvoll. Das Ziel besteht darin, die verschiedenen Maßnahmen einander gegenüberzustellen. Während bei den Umsetzungen vor allem auf die Gewährleistung der Integrität und Vertraulichkeit von Daten sowie die Minimierung der Angriffsfläche eingegangen wurde, sollen hier vor allem die negativen Einflüsse ermittelt werden. Grundlage dafür ist die in der Methodik beschriebene Vorgehensweise. Dabei stehen die Herausforderungen der Implementierung und die Auswirkungen auf die Performance des Systems im Vordergrund.

Zur Ermittlung der zusätzlichen Programmlaufzeit wurden die aus der Umsetzung bekannten Szenarien genutzt, um eine Zeiterfassung vorzunehmen. Diese Szenarien wurden mit Beispielen unterschiedlicher Größe durchgeführt. Dabei kamen TIFF-Bilder mit 101 MB sowie 191 KB und BMP-Bilder mit 75,8 MB und 92,9 KB zum Einsatz. Die Bilder dienten als Versuchsobjekte für die Durchführung der Tests zur Kapselung und Verschlüsselung. Dadurch lassen sich die Durchschnittszeit, die Maximalzeit, die Durchschnitt-Zusatzzeit und die Maximal-Zusatzzeit ermitteln. Die Zeitmessung erfolgte durch den Einsatz der Windows-API-Funktion `QueryPerformanceCounter`, welche mit einer Genauigkeit von unter einer Mikrosekunde ($<1 \mu\text{s}$) Messungen ermöglicht und somit die Erfassung der Ausführungszeit ermöglicht. [Microsoft, 2025m] Der dafür verwendete Code zur Zeitmessung ist in Abbildung 5.1 dargestellt.

```
// Zeitmessung start
double timeSpend = 0;
double pcFreq = 0.0;
__int64 counterStart = 0;
LARGE_INTEGER li_start;
LARGE_INTEGER li_end;

QueryPerformanceFrequency(&li_start);
pcFreq = double(li_start.QuadPart)/1000.0;
QueryPerformanceCounter(&counterStart);

// Zeitmessung ende
QueryPerformanceCounter(&li_end);
timeSpend = double(li_end.QuadPart-counterStart)/pcFreq;
```

Abbildung 5.1.: Codebeispiel zur Zeitmessung mit QueryPerformanceCounter

Um eine konsistente Zeiterfassung zu gewährleisten, wurden die Tests unter denselben Bedingungen mit dem folgenden Testsystem durchgeführt:

- Betriebssystem: Windows 10 Pro 22H2
- Prozessor: AMD Ryzen 5 5600X mit 3,70 GHz
- Grafikkarte: Nvidia GeForce RTX 3060 Ti
- RAM: 32 GB DDR4
- Festplatte: NVMe Samsung SSD 970

Für jede getestete Sicherheitsmaßnahme wurden 50 Testdurchläufe durchgeführt. Die detaillierten Ergebnisse der Verschlüsselungsmaßnahmen sind im Anhang A.4 zu finden, die der Kapselungsmaßnahmen im Anhang A.5. In der nachfolgenden Tabelle 5.1 sind die Ergebnisse zur Programmlaufzeit für das in Abbildung 4.7 dargestellte Bild sowie die Bewertungen der in Kapitel 3.4 beschriebenen Faktoren zusammengefasst. Die Bewertungen folgen dabei dem aufgeführten numerischen Punktesystem von 1 bis 10, wodurch die Tabelle eine übersichtliche Einstufung der verschiedenen Schutzmaßnahmen darstellt und somit als Entscheidungsgrundlage für die Einführung zukünftiger Schutzmaßnahmen dient. Zu beachten ist, dass die anfallenden Kosten in dieser Tabelle nicht aufgeführt werden, da FIDO2 mit durchschnittlich 35,30 € pro Hardware-Token die einzige kostenerzeugende Maßnahme ist. Der FIDO2-Ansatz wurde nur theoretisch betrachtet und basiert

auf Schätzungen, die nicht im Rahmen einer praktischen Umsetzung verifiziert wurden. Deshalb wird diese Schutzmaßnahme auch nicht in der Tabelle aufgeführt. Für die berücksichtigten Faktoren gilt, dass die Skalenwerte anhand der folgenden Definitionen für jeden Faktor interpretiert werden können:

- **Zusätzliche Programmlaufzeit:** Dieser Faktor beurteilt die durch die Schutzmaßnahme zusätzlich benötigte Programmlaufzeit im Vergleich zur Ausführung ohne Maßnahme. In Anlehnung an die Benutzerfreundlichkeit beim Laden von Webseiten wird eine zusätzliche Laufzeit von bis zu zwei Sekunden als maximaler Wert von 10 betrachtet. [Stringam and Gerdes, 2019] Die benötigte Zusatzzeit wurde basierend auf den detaillierten Ergebnissen prozentual erfasst und anschließend auf die Skala von 1 bis 10 übertragen.
- **Erforderliche Vorkenntnisse:** Hierbei handelt es sich um eine Einschätzung des erforderlichen Wissens, um die Maßnahme erfolgreich umzusetzen. Diese Einschätzung basiert auf der durchgeführten Forschung und Vorbereitung. Ein Wert von 1 signalisiert einen geringen Bedarf an besonderen Kenntnissen, während ein Wert von 10 auf einen hohen Bedarf hinweist.
- **Zeitaufwand:** Dieser Faktor misst den erforderlichen Zeitaufwand von der Einarbeitung bis zur prototypischen Umsetzung der Maßnahme. Dabei wird ein Tag als minimaler Aufwand mit dem Wert 1 festgelegt und zwei Monate oder mehr als maximaler Aufwand mit dem Wert 10.
- **Schwierigkeit der Implementierung:** Diese Bewertung basiert auf der Einschätzung der Schwierigkeit bei der Umsetzung der Schutzmaßnahme, die während der prototypischen Implementierung beobachtet wurde. Ein höherer Wert weist auf eine größere Komplexität hin.
- **Nutzen der Maßnahme:** Dieser Faktor beurteilt, wie effektiv die Maßnahme ist, um die im Theorieteil vorgestellten Angriffsmöglichkeiten vorzubeugen. Die Einschätzung erfolgt auf Grundlage der theoretischen Wirksamkeit der Maßnahme, wobei höhere Werte für einen besseren Schutz stehen.

Maßnahme	Durchschnittlicher Laufzeitverlust	Erforderliche Vorkenntnisse	Zeitaufwand	Schwierigkeit	Nutzen der Maßnahme
XOR Verschlüsselung	2	1	1	1	1
AES-256-CBC Verschlüsselung	4	4	3	3	5
Einfache Kapselung	7	4	3	4	2
Verschlüsselte Kapselung	10	7	6	7	4
IPC Kapselung	4	5	4	4	6
Sandbox Kapselung	4	9	8	9	8

Tabelle 5.1.: Zusammengefasste Ergebnisse und Bewertungen der Sicherheitsmaßnahmen

6. Diskussion

In diesem Kapitel werden die Umsetzung und Bewertung der verschiedenen Sicherheitsmaßnahmen reflektiert. Der Fokus liegt dabei auf Implementierungsempfehlungen, identifizierten Herausforderungen und verbleibenden Sicherheitslücken.

Die untersuchten Sicherheitsmaßnahmen umfassen die Bereiche Verschlüsselung, Authentifizierung und Kapselung, die jeweils auf unterschiedliche Sicherheitsanforderungen abzielen. Verschlüsselungsmethoden wie AES-256-CBC adressieren das grundlegende Sicherheitsproblem der Datenvertraulichkeit, indem sie den Zugriff auf sensible Informationen durch Unbefugte erschweren. Dieses Vorgehen entspricht der Anforderung der DSGVO. Erweiterte Authentifizierungsverfahren wie FIDO2 bieten Schutzmechanismen gegen Phishing-Angriffe, indem sie sicherstellen, dass nur autorisierte Benutzer Zugang zu Systemen und sensiblen Daten erhalten. Dabei wird die Authentizität eines Benutzers gestärkt, indem zusätzlich zur Authentifizierung ein besitzbasierter Faktor, etwa ein USB-Token, in Kombination mit einer persönlichen Identifikationsnummer verlangt wird. Um sich erfolgreich als dieser Benutzer auszugeben, ist sowohl der direkte Diebstahl des Tokens als auch die Kenntnis der zugehörigen PIN nötig. Auf diese Weise werden die Anforderungen der ISO 27001 erfüllt. Kapselungstechniken erfüllen eine vielseitige Schutzfunktion. Sie stärken die Daten- und Prozessintegrität, indem sie bestimmte Funktionen isolieren. Beim Sandboxing werden potenziell riskante Aktivitäten beispielsweise in einer kontrollierten Umgebung ausgeführt, wodurch das Gesamtsystem geschützt wird. Im Gegensatz dazu zielt die Kapselung mittels Interprozesskommunikation darauf ab, den Hauptprozess der Anwendung zu schützen. Dazu werden Drittanbieterbibliotheken isoliert und deren potenzielle Schwachstellen eingeschränkt.

Der Aufwand für Dokumentation und Einarbeitung variiert je nach Maßnahme. Ein grundlegendes Verständnis der Funktionsweise der verwendeten Verschlüsselungsmethoden und Authentifizierungsverfahren ist zwar wichtig. OWASP empfiehlt jedoch, auf bewährte Bibliotheken von Drittanbietern zurückzugreifen. [OWASP, 2025] Der Grund dafür ist, dass

bei einer Eigenimplementierung ein hohes Maß an Fachkenntnissen sowie eine präzise Umsetzung erforderlich sind. Zudem müssen jegliche Schwachstellen berücksichtigt sowie regelmäßige Wartungen und Prüfungen auf neue Angriffsmöglichkeiten durchgeführt werden. Bei der Verwendung von Drittanbieterbibliotheken sind ebenfalls regelmäßige Wartungen und Prüfungen auf Sicherheitslücken erforderlich, der Implementierungsaufwand ist jedoch vertretbar. Zusätzlich können bestimmte Funktionen durch Kapselungstechniken isoliert werden.

Auch in Bezug auf die Performance gibt es Unterschiede. So kann die AES-Verschlüsselung aufgrund der wiederholten Verschlüsselungs- und Entschlüsselungsvorgänge je nach Dateigröße viel Leistung beanspruchen. Authentifizierungsverfahren wie FIDO2, die in der Regel nur beim Anmeldeprozess ausgeführt werden, belasten das System dagegen weniger. Kapselungstechniken ohne den Einsatz von Interprozesskommunikation sind hingegen nicht empfehlenswert. Grund dafür ist die umständliche Kommunikation beim Speichern und Lesen von Dateien, was zu unvermeidbaren Verzögerungen führt. Empfohlen wird deshalb die Sandbox-Implementierung, da sie umfassende Schutzmaßnahmen bietet und gleichzeitig nur akzeptable Performanceverluste verursacht.

Die Anpassbarkeit dieser Sicherheitsmaßnahmen an verschiedene Aufgabenstellungen zeigt ihre Vielseitigkeit. Verschlüsselung ist vorteilhaft, wenn sensible Daten auf einem System gespeichert oder über das Internet übertragen werden müssen. Eine erweiterte Authentifizierung ist sinnvoll, um sicherzustellen, dass nur berechtigte Personen Zugriff erhalten. Die Kapselung ist besonders nützlich, wenn einmalige Funktionen von Bibliotheken externer Anbieter benötigt werden. In diesem Fall stellt die Sandbox-Implementierung eine überlegene Lösung dar, da sie nicht nur isoliert, sondern das gesamte System auch vor schädlichen Aufrufen schützt. Was sogar in Internetbrowsern wie Chrome Anwendung findet.

Allerdings sind innerhalb der genannten Schutzmaßnahmen konzeptbedingte Schwachstellen vorhanden, die wie folgt aufgezeigt werden:

- XOR-Verschlüsselung: Diese Verschlüsselung gilt als unsicher, da sie sich durch einfache Kryptoanalyse leicht entschlüsseln lässt und somit nicht zum Schutz sensibler Daten geeignet ist.

- AES-CBC-Verschlüsselung: Kann durch Bit-Flipping-Angriffe gefährdet werden. Dabei werden Bits im Chiffretext manipuliert, um den resultierenden Klartext gezielt zu verändern und somit die Integrität der Daten anzugreifen. [Jadaan, 2025]
- FIDO2-Authentifizierung: Die Anforderung einer Internetverbindung eröffnet eine Vielzahl an Sicherheitslücken, beispielsweise die Möglichkeit von Cross-Site-Request-Forgery-Angriffen, bei denen ein Nutzer unwissentlich Aktionen ausführt, die er nicht autorisiert hat. Außerdem sind spezielle Wiederherstellungsstrategien notwendig, um verlorene USB-Tokens zu ersetzen, da diese einzigartig und nicht duplizierbar sind. [BSI, 2025d]
- Kapselung mittels Interprozesskommunikation: Es besteht die Anfälligkeit für Man-in-the-Middle-Angriffe, bei denen Kommunikationsdaten zwischen Prozessen abgefangen werden können.
- Kapselung mit verschlüsselter Speicherung: Problematisch ist die Übergabe von Schlüsseln und Initialisierungsvektoren über die Kommandozeile an einen Prozess. Dadurch können Angreifer den Schlüssel und den IV ebenfalls durch Man-in-the-Middle-Angriffe abfangen und verwenden, um die verschlüsselten Informationen zu entschlüsseln.
- Kapselung mit Sandbox: Hierbei besteht das Risiko eines sogenannten Sandbox-Ausbruchs. Dabei nutzt ein Angreifer Sicherheitslücken der Sandbox aus, um auszubrechen und Zugriff auf das darunterliegende System oder andere geschützte Bereiche zu erlangen. [Or, 2025][Sangfor, 2025]

Abschließend wird auf Basis der untersuchten Sicherheitsmaßnahmen und deren Evaluierung die Implementierung der Sandboxing-Schutzmaßnahme empfohlen. Sandboxing bietet umfangreiche Vorteile, da potenziell riskante Aktivitäten in einer kontrollierten Umgebung isoliert werden. Dadurch wird die Integrität und Sicherheit des Gesamtsystems erhöht und das Risiko, dass schädliche Aktionen direkten Einfluss auf das System nehmen können, wird verringert. Zudem zeichnet sich die Sandbox-Implementierung dadurch aus, dass sie neben den Sicherheitsvorteilen nur akzeptable Performanceverluste verursacht und flexibel an verschiedene Anwendungsbereiche angepasst werden kann.

7. Ausblick

Im abschließenden Kapitel soll ein Ausblick auf die mögliche Zukunft des behandelten Themenbereichs erfolgen. Das Ziel besteht darin, ein Verständnis dafür zu vermitteln, wo weitere Forschungen sinnvoll ansetzen können. Zu diesem Zweck wird noch einmal aufgezeigt, in welchem Rahmen die Ergebnisse dieser Arbeit gelten, welche weiteren Maßnahmen möglich sind, wie diese Arbeit in der Forschung einzuordnen ist, welche Erkenntnisse aus der Arbeit mitzunehmen sind und wie die Zukunft dieses Themenbereichs aussehen könnte.

In der vorliegenden Arbeit wurde untersucht, wie sich die Cybersicherheit von Software als Medizinprodukt verbessern lässt. Zur Beantwortung dieser Frage wurden die zugrundeliegenden Gesetze, Richtlinien, Normen und Verordnungen analysiert. Auf deren Grundlage konnten anschließend geeignete Sicherheitsmaßnahmen abgeleitet werden. Anhand der behandelten Maßnahmen wurde dargelegt, wie sich Verschlüsselung, Authentifizierung und Kapselungstechniken nutzen lassen, um die Sicherheit zu erhöhen.

Nach der Untersuchung der Sicherheitsmaßnahmen stellt sich die Frage, welche Schritte unternommen werden können, um die Cybersicherheit weiter zu erhöhen. Einerseits sollten Ansätze untersucht werden, wie sich die konzeptbedingten Schwachstellen der bestehenden Sicherheitsmaßnahmen schließen lassen. Andererseits wurde der Themenbereich der Verifikation mithilfe von Werkzeugen aufgrund einer bereits vorhandenen Umsetzung nicht behandelt. Softwarelösungen, die diese Schutzmaßnahme nicht berücksichtigen, sollten jedoch Softwarestücklisten erforschen. Denn Softwarestücklisten sind ein wichtiges Mittel, um die bestehenden Abhängigkeiten innerhalb einer Software zu erfassen und somit Wartungsarbeiten zu vereinfachen. In Kombination mit der CVE-Datenbank lassen sich so zusätzlich bekannte Sicherheitslücken identifizieren. Des Weiteren bieten Schritte aus der Chromium-Sandbox, wie verschiedene Prozessmitigationsrichtlinien, die Möglichkeit, die Sandbox widerstandsfähiger zu machen. Eine Analyse der Dokumentation der Chromium-Sandbox ist eine geeignete Grundlage, um Verbesserungen zu realisieren. Dar-

über hinaus gewinnt die Nutzung von maschinellem Lernen zunehmend an Aufmerksamkeit. Eine grundlegende Recherche zu den Möglichkeiten, Sicherheitslücken mit maschinellem Lernen zu erkennen, würde demnach neue Potenziale aufzeigen.

Um die Ergebnisse sinnvoll in den bestehenden Forschungskontext einzuordnen, folgt eine Einschätzung ihrer Relevanz. Die vorliegende Arbeit bietet eine wertvolle Grundlage, Einblicke in die Anforderungen der Cybersicherheit im Bereich der Software für Medizinprodukte zu erlangen. Zwar werden grundlegende Umsetzungsmöglichkeiten aufgezeigt, eine tiefere Betrachtung der einzelnen Maßnahmen bleibt jedoch aus, da dies den Rahmen dieser Arbeit bei Weitem überschreiten würde. Aus demselben Grund konnten auch andere Sicherheitsmaßnahmen nicht berücksichtigt oder umgesetzt werden. Die Untersuchung diene demnach hauptsächlich der prototypischen Darstellung und war durch die Anzahl der evaluierten Sicherheitsmechanismen begrenzt. Künftige Arbeiten sollten weitere Sicherheitslösungen evaluieren und einzelne Implementierungen vertiefen.

Der Leser sollte mitnehmen, dass Cybersicherheit ein dynamisches Feld ist, welches kontinuierliche Anpassung und Flexibilität erfordert. Sicherheitsmaßnahmen sind keine einmaligen Installationen, sondern müssen als fortlaufender Prozess betrachtet werden, der technische und organisatorische Aspekte einschließt. Die Entwicklung fortschrittlicherer Technologien wie Quantencomputer wird nämlich neue Angriffsszenarien schaffen. Obwohl Quantencomputer schnellere Berechnungen ermöglichen, besteht auch die Gefahr, dass bestehende Verschlüsselungsverfahren hinfällig werden. Die Entwicklung quantensicherer Verschlüsselungsmethoden wird daher unerlässlich sein, um die Zukunft der Datensicherheit zu ermöglichen. Das Ziel muss darin bestehen, sich den wandelnden Bedrohungen der digitalen Welt wirksam zu stellen. Zukünftige Forschung und Weiterentwicklung sind daher unerlässlich, um bestehende und kommende Herausforderungen erfolgreich zu bewältigen.

Literaturverzeichnis

- [Alfes et al., 2018] Alfes, J., Schaefer, I., and Kolip, P. (2018). Voraussetzungen für einen erfolgreichen Projekttransfer. *Das Gesundheitswesen*, 80(02):144–148.
- [Alslman et al., 2023] Alslman, Y. S., Ahmad, A., and AbuHour, Y. (2023). Enhanced and authenticated cipher block chaining mode. *Bulletin of Electrical Engineering and Informatics*, 12(4):2357–2362.
- [Altenhofer and Studer, 2011] Altenhofer, W. and Studer, L. (2011). *TPM-basierter System Health Agent unter Windows 7-Trusted NAP*. PhD thesis, HSR Hochschule für Technik Rapperswil.
- [BACS, 2024] BACS (2024). Bundesamt für Cybersicherheit Schweiz - Aktuelle Zahlen — ncsc.admin.ch. <https://www.ncsc.admin.ch/ncsc/de/home/aktuell/aktuelle-zahlen.html>. [Online; Stand 16.12.2024].
- [Barker et al., 2007] Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2007). Recommendation for Key Management - Part 1: General (Revised) (March 2007 edition).
- [Beiersmann, 2025] Beiersmann, S. (2025). Forscher decken Sicherheitslücken in TPM-2.0-Chips auf | ZDNet.de — zdnet.de. <https://www.zdnet.de/88407546/forscher-decken-sicherheitsluecken-in-tpm-2-0-chips-auf/>. [Online; Stand 31.03.2025].
- [BfArM, 2024] BfArM (2024). Bundesinstitut für Arzneimittel und Medizinprodukte - Abgrenzung/Klassifizierung — bfarm.de. https://www.bfarm.de/DE/Medizinprodukte/Aufgaben/Abgrenzung-und-Klassifizierung/_node.html. [Online; Stand 15.12.2024].
- [BMG, 2025] BMG (2025). Bundesministerium für Gesundheit - Was sind Medizinprodukte? [Online; Stand 20.05.2025].
- [Bourke, 1998] Bourke, P. (1998). Bmp image format. *BMP Files*. July, 8.

- [BSI, 2024a] BSI (2024a). Bundesamt für Sicherheit in der Informationstechnik - Die Lage der IT-Sicherheit in Deutschland — bsi.bund.de. https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Lagebericht/lagebericht_node.html. [Online; Stand 16.12.2024].
- [BSI, 2024b] BSI (2024b). Bundesamt für Sicherheit in der Informationstechnik - EU-Richtlinien zur Netzwerk- und Informationssicherheit — bsi.bund.de. https://www.bsi.bund.de/DE/Das-BSI/Auftrag/Gesetze-und-Verordnungen/NIS-Richtlinien/nis-richtlinie_node.html. [Online; Stand 12.12.2024].
- [BSI, 2025a] BSI (2025a). Bundesamt für Sicherheit in der Informationstechnik — bsi.bund.de. https://www.bsi.bund.de/DE/Home/home_node.html. [Online; Stand 12.02.2025].
- [BSI, 2025b] BSI (2025b). Bundesamt für Sicherheit in der Informationstechnik - Cyber Resilience Act — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Cyber_Resilience_Act/cyber_resilience_act_node.html. [Online; Stand 30.01.2025].
- [BSI, 2025c] BSI (2025c). Bundesamt für Sicherheit in der Informationstechnik - Der Browser – Gefahren und Risiken — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Updates-Browser-Open-Source-Software/Der-Browser/Gefahren-und-Risiken/gefahren-und-risiken_node.html. [Online; Stand 12.02.2025].
- [BSI, 2025d] BSI (2025d). Bundesamt für Sicherheit in der Informationstechnik - Die Kryptografie hinter Passkey — bsi.bund.de. <https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Accountschutz/Passkeys/Kryptografie-hinter-Passkey/kryptografie-hinter-passkey.html>. [Online; Stand 22.05.2025].
- [BSI, 2025e] BSI (2025e). Bundesamt für Sicherheit in der Informationstechnik - DoS- und DDoS-Attacken — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/DoS-Denial-of-Service/dos-denial-of-service_node.html. [Online; Stand 12.02.2025].

- [BSI, 2025f] BSI (2025f). Bundesamt für Sicherheit in der Informationstechnik - Geltungsbereich Lauschabwehr im Bereich der Wirtschaft — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Zertifizierung-und-Anerkennung/Anerkennung-von-Stellen-und-Zertifizierung-IT-Sicherheitsdienstleister/Lauschabwehr-im-Bereich-der-Wirtschaft/lauschabwehr-im-bereich-der-wirtschaft_node.html. [Online; Stand 12.02.2025].
- [BSI, 2025g] BSI (2025g). Bundesamt für Sicherheit in der Informationstechnik - Java-Bibliothek Log4j – eine Bilanz — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Schwachstelle-log4Shell-Java-Bibliothek/log4j_node.html. [Online; Stand 02.05.2025].
- [BSI, 2025h] BSI (2025h). Bundesamt für Sicherheit in der Informationstechnik - Malware — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Gefahren/Malware/malware_node.html. [Online; Stand 12.02.2025].
- [BSI, 2025i] BSI (2025i). Bundesamt für Sicherheit in der Informationstechnik - Phishing - how much is the phish!? — bsi.bund.de. https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Spam-Phishing-Co/Passwortdiebstahl-durch-Phishing/passwortdiebstahl-durch-phishing_node.html. [Online; Stand 12.02.2025].
- [BSI, 2025j] BSI (2025j). Bundesamt für Sicherheit in der Informationstechnik - Relationale Datenbanken — bsi.bund.de. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-GS-Kompendium_Einzel_PDFs_2023/06_APP_Anwendungen/APP_4_3_Relationale_Datenbanksysteme_Edition_2023.pdf?__blob=publicationFile&v=3. [Online; Stand 12.02.2025].
- [BSI, 2025k] BSI (2025k). Bundesamt für Sicherheit in der Informationstechnik - TR-02102-1 "Kryptographische Verfahren: Empfehlungen und Schlüssellängen" Version: 2025-01 — bsi.bund.de. <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.html>. [Online; Stand 01.06.2025].
- [BT, 2024] BT (2024). Deutscher Bundestag - Gesetze — bundestag.de. <https://www.bundestag.de>.

- w.bundestag.de/services/glossar/glossar/G/gesetze-245434. [Online; Stand 14.12.2024].
- [Buell, 2024] Buell, D. (2024). Asymmetrische Chiffren–RSA und andere. In *Grundlagen der Kryptographie: Einführung in die mathematischen und algorithmischen Grundlagen*, pages 165–172. Springer.
- [Christiane Toyka-Seid, 2024] Christiane Toyka-Seid, G. S. (2024). Verordnung / Rechtsverordnung — bpb.de. <https://www.bpb.de/kurz-knapp/lexika/das-junge-politik-lexikon/321319/verordnung-rechtsverordnung/>. [Online; Stand 11.12.2024].
- [CVE, 2025] CVE (2025). Common Vulnerabilities and Exposures — cve.org. <https://www.cve.org/>. [Online; Stand 23.03.2025].
- [Dehning, 2024] Dehning, O. (2024). Transparenz in der IT-Supply-Chain: SBOM-Software Bill of Materials. *Datenschutz und Datensicherheit-DuD*, 48(1):25–30.
- [Demant, 2018] Demant, C. (2018). Der Quellcode.
- [Descope, 2025] Descope (2025). WebAuthn: How it Works & Example Flows — descope.com. <https://www.descope.com/learn/post/webauthn>. [Online; Stand 19.05.2025].
- [DIN, 2003] DIN (2003). Deutsche Institut für Normung e.V. - Medizingeräte-Software - Software-Lebenszyklus-Prozesse.
- [DIN, 2015] DIN (2015). Deutsche Institut für Normung e.V. - Medizinprodukte - Teil 1: Anwendung der Gebrauchstauglichkeit auf Medizinprodukte.
- [DIN, 2019] DIN (2019). Deutsche Institut für Normung e.V. - Medizinprodukte - Anwendung des Risikomanagements auf Medizinprodukte.
- [DIN, 2021] DIN (2021). Deutsche Institut für Normung e.V. - Medizinprodukte - Qualitätsmanagementsysteme - Anforderungen für regulatorische Zwecke.
- [DIN, 2022] DIN (2022). Deutsche Institut für Normung e.V. - Medizinprodukte - Anwendung des Risikomanagements auf Medizinprodukte.
- [DIN, 2024] DIN (2024). Deutsche Institut für Normung e.V. - Informationssicherheit, Cybersicherheit und Datenschutz - Informationssicherheitsmanagementsysteme - Anforderungen.

- [DIN, 2025] DIN (2025). Deutsche Institut für Normung e.V. - Normungsprozess: So entsteht eine Norm — [din.de. https://www.din.de/de/ueber-normen-und-standards/din-norm/normungsprozess](https://www.din.de/de/ueber-normen-und-standards/din-norm/normungsprozess). [Online; Stand 30.01.2025].
- [Eichstädt and Spieker, 2024] Eichstädt, T. and Spieker, S. (2024). Authentifizierung. In *52 Stunden Informatik: Was jeder über Informatik wissen sollte*, pages 93–98. Springer.
- [Embarcadero, 2025a] Embarcadero (2025a). Embarcadero Technologies, Inc - System.Classes.TFileStream 2013; RAD Studio API Documentation — [docwiki.embarcadero.com. https://docwiki.embarcadero.com/Libraries/Sydney/de/System.Classes.TFileStream](https://docwiki.embarcadero.com/Libraries/Sydney/de/System.Classes.TFileStream). [Online; Stand 08.04.2025].
- [Embarcadero, 2025b] Embarcadero (2025b). Embarcadero Technologies, Inc - System.Classes.TStream 2013; RAD Studio API Documentation — [docwiki.embarcadero.com. https://docwiki.embarcadero.com/Libraries/Sydney/de/System.Classes.TStream](https://docwiki.embarcadero.com/Libraries/Sydney/de/System.Classes.TStream). [Online; Stand 08.04.2025].
- [Ernst et al., 2023] Ernst, H., Schmidt, J., and Beneken, G. (2023). Verschlüsselung.
- [EU, 2017] EU (2017). Europäische Union - Verordnung - 2017/745 - EN - EUR-Lex — [eur-lex.europa.eu. https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX%3A32017R0745](https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX%3A32017R0745). [Online; Stand 14.12.2024].
- [EU, 2024] EU (2024). Europäischen Union - Arten von Rechtsvorschriften — [european-union.europa.eu. https://european-union.europa.eu/institutions-law-budget/law/types-legislation_de](https://european-union.europa.eu/institutions-law-budget/law/types-legislation_de). [Online; Stand 11.12.2024].
- [EU, 2025a] EU (2025a). Europäische Kommission - MDCG 2021-24 Guidance on classification of medical devices. https://health.ec.europa.eu/medical-devices-sector/new-regulations/guidance-mdcg-endorsed-documents-and-other-guidance_en. [Online; Stand 03.01.2025].
- [EU, 2025b] EU (2025b). Europäische union - Kapitel 2 – Grundsätze — [dsgvo-gesetz.de. https://dsgvo-gesetz.de/kapitel-2/](https://dsgvo-gesetz.de/kapitel-2/). [Online; Stand 02.01.2025].
- [FIDO, 2019] FIDO (2019). FIDO-Allianz - Client to Authenticator Protocol (CTAP) — [fidoalliance.org. https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html](https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html). [Online; Stand 17.05.2025].

- [FIDO, 2025] FIDO (2025). FIDO-Allianz - FIDO2 — fidoalliance.org. <https://fidoalliance.org/fido2/>. [Online; Stand 17.05.2025].
- [FIDO, 2025] FIDO (2025). FIDO-Allianz - GitHub - fido-alliance/webauthn-demo: WebAuthn Workshop Demo [Completed DEMO is at completed demo-branch] — github.com. <https://github.com/fido-alliance/webauthn-demo>. [Online; Stand 18.05.2025].
- [Fox, 2009] Fox, D. (2009). Sicheres löschen von Daten auf Festplatten. *Datenschutz und Datensicherheit-DuD*, 33(2):110–113.
- [Gärtner, 2010] Gärtner, A. (2010). Einsatz und Betrieb von Software als Medizinprodukt.
- [Google, 2025] Google (2025). Chromium Docs - Sandbox — chromium.googlesource.com. [https://chromium.googlesource.com/chromium/src/+main/docs/design/sandbox.md](https://chromium.googlesource.com/chromium/src/+/main/docs/design/sandbox.md). [Online; Stand 19.03.2025].
- [Hastenteufel et al., 2019] Hastenteufel, M., Renaud, S., Hastenteufel, M., and Renaud, S. (2019). Software als Medizinprodukt. *Software als Medizinprodukt: Entwicklung und Zulassung von Software in der Medizintechnik*, pages 7–178.
- [IHK Rhein-Neckar, 2024] IHK Rhein-Neckar (2024). Industrie- und Handelskammer Rhein-Neckar - Definition Normen – Standards — ihk.de. <https://www.ihk.de/rhein-neckar/innovation/innovationsberatung/normen-normung/definition-normen-standards-5551358>. [Online; Stand 12.12.2024].
- [Image Instruments, 2024a] Image Instruments (2024a). Image Instruments GmbH - Installation — onyxwiki.net. <https://www.onyxwiki.net/doku.php?id=installation>. [Online; Stand 18.10.2024].
- [Image Instruments, 2024b] Image Instruments (2024b). Image Instruments GmbH - Modules — onyxwiki.net. https://www.onyxwiki.net/doku.php?id=modules_000. [Online; Stand 29.12.2024].
- [Image Instruments, 2024c] Image Instruments (2024c). Image Instruments GmbH - Start — onyxwiki.net. <https://www.onyxwiki.net/doku.php?id=start>. [Online; Stand 18.12.2024].

- [Image Instruments, 2024d] Image Instruments (2024d). Image Instruments GmbH - Systemrequirements — onyxwiki.net. <https://www.onyxwiki.net/doku.php?id=systemrequirements>. [Online; Stand 29.12.2024].
- [IMDRF, 2013] IMDRF (2013). Software as a Medical Device (SaMD): Key Definitions. International Medical Device Regulators Forum.
- [Jadaan, 2025] Jadaan, M. (2025). CBC Bit-Flipping Attack — masjadaan. <https://medium.com/@masjadaan/cbc-bit-flipping-attack-b92efafbb83d>. [Online; Stand 22.05.2025].
- [Johner Institut, 2025] Johner Institut (2025). Johner Institut GmbH - IT Security — johner-institut.de. <https://www.johner-institut.de/blog/tag/it-security/>. [Online; Stand 02.01.2025].
- [Kanidm, 2025] Kanidm (2025). Kanidm Identity Management Project - GitHub - kanidm/webauthn-rs: An implementation of webauthn components for Rustlang servers — github.com. <https://github.com/kanidm/webauthn-rs>. [Online; Stand 18.05.2025].
- [Karpfinger and Kiechle, 2010] Karpfinger, C. and Kiechle, H. (2010). *Das RSA-Verfahren*, pages 111–140. Vieweg+Teubner, Wiesbaden.
- [Keschull, 2023] Keschull, U. (2023). Allgemeine Maßnahmen.
- [Kneuper, 2021] Kneuper, R. (2021). *Datenschutz für Softwareentwicklung und IT*. Springer.
- [Krebs and Hagenweiler, 2022] Krebs, H.-A. and Hagenweiler, P. (2022). Verfahren zur Durchführung der Anonymisierung.
- [Lenhard, 2020] Lenhard, T. H. (2020). Verschlüsselung.
- [LibTIFF, 2025] LibTIFF (2025). TIFF Library and Utilities — LibTIFF 4.7.0 documentation — libtiff.gitlab.io. <https://libtiff.gitlab.io/libtiff/>. [Online; Stand 06.05.2025].
- [Liebe et al., 2024] Liebe, A., Kroon, P., and Wiewiorra, L. (2024). Data Access, Data Sharing und Privacy. WIK Diskussionsbeitrag 527, Bad Honnef.

- [Lukeman, 2025] Lukeman, R. (2025). Foto von Robert Lukeman auf Unsplash — unsplash.com. <https://unsplash.com/de/fotos/wasserfalle-unter-bewolktem-himmel-PH0HYjsf2n8>. [Online; Stand 16.04.2025].
- [Mandl, 2014] Mandl, P. (2014). Prozesse und Threads.
- [Mason, 2005] Mason, S. (2005). Trusted computing and forensic investigations. *Digital Investigation*, 2(3):189–192.
- [Meinel et al., 2014] Meinel, C., Sack, H., Meinel, C., and Sack, H. (2014). Vertraulichkeit und Verschlüsselung.
- [Microsoft, 2025a] Microsoft (2025a). Microsoft Corporation - Auftragsobjekte - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/procthread/job-objects>. [Online; Stand 12.05.2025].
- [Microsoft, 2025b] Microsoft (2025b). Microsoft Corporation - Bitmaptypen - Windows Forms .NET Framework — learn.microsoft.com. <https://learn.microsoft.com/de-de/dotnet/desktop/winforms/advanced/types-of-bitmaps?view=netframeworkdesktop-4.8>. [Online; Stand 07.04.2025].
- [Microsoft, 2025c] Microsoft (2025c). Microsoft Corporation - CPU-Analyse — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows-hardware/test/wpt/cpu-analysis>. [Online; Stand 03.05.2025].
- [Microsoft, 2025d] Microsoft (2025d). Microsoft Corporation - CreateProcessAsUserW-Funktion (processthreadsapi.h) - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessasuserw>. [Online; Stand 12.05.2025].
- [Microsoft, 2025e] Microsoft (2025e). Microsoft Corporation - CreateProcessW-Funktion (processthreadsapi.h) - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessw>. [Online; Stand 07.05.2025].
- [Microsoft, 2025f] Microsoft (2025f). Microsoft Corporation - CreateRestrictedToken-Funktion (securitybaseapi.h) - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/api/securitybaseapi/nf-securitybaseapi-createrestrictedtoken>. [Online; Stand 12.05.2025].

- [Microsoft, 2025g] Microsoft (2025g). Microsoft Corporation - DACLs und ACEs - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/secauthz/daccls-and-aces>. [Online; Stand 12.05.2025].
- [Microsoft, 2025h] Microsoft (2025h). Microsoft Corporation - Eingeschränkte Token - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/secauthz/restricted-tokens>. [Online; Stand 12.05.2025].
- [Microsoft, 2025i] Microsoft (2025i). Microsoft Corporation - Funktionsweise eines Streams — learn.microsoft.com. <https://learn.microsoft.com/de-de/cpp/standard-library/what-a-stream-is?view=msvc-170>. [Online; Stand 08.04.2025].
- [Microsoft, 2025j] Microsoft (2025j). Microsoft Corporation - Interprocess communications - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/ipc/interprocess-communications>. [Online; Stand 10.05.2025].
- [Microsoft, 2025k] Microsoft (2025k). Microsoft Corporation - Microsoft Entra ID- Nachweis für Anbieter von FIDO2-Sicherheitsschlüsseln — learn.microsoft.com. <https://learn.microsoft.com/de-de/entra/identity/authentication/concept-fido2-hardware-vendor>. [Online; Stand 18.05.2025].
- [Microsoft, 2025l] Microsoft (2025l). Microsoft Corporation - Obligatorische Integritätssteuerung - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/secauthz/mandatory-integrity-control>. [Online; Stand 12.05.2025].
- [Microsoft, 2025m] Microsoft (2025m). Microsoft Corporation - QueryPerformanceCounter-Funktion - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/api/profileapi/nf-profileapi-queryperformancecounter>. [Online; Stand 20.05.2025].
- [Microsoft, 2025n] Microsoft (2025n). Microsoft Corporation - Systemfehlercodes (0-499) (WinError.h) - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/debug/system-error-codes--0-499->. [Online; Stand 01.06.2025].
- [Microsoft, 2025o] Microsoft (2025o). Microsoft Corporation - Systemfehlercodes (1700-3999) (WinError.h) - Win32 apps — learn.microsoft.com. <https://learn.microsoft.com/de-de/windows/win32/debug/system-error-codes--1700-3999->. [Online; Stand 01.06.2025].

- t.com/de-de/windows/win32/debug/system-error-codes--1700-3999-. [Online; Stand 01.06.2025].
- [Microsoft, 2025p] Microsoft (2025p). Microsoft Corporation - Understanding the Active Directory Logical Model — learn.microsoft.com. <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/understanding-the-active-directory-logical-model>. [Online; Stand 19.03.2025].
- [Müller, 2008] Müller, T. (2008). Trusted-Computing-Systeme.
- [Nehrke, 2023] Nehrke, L. (2023). Webdienst zur Überwachung von Schwachstellen in Software-Stücklisten (SBOM).
- [Oodrive, 2025] Oodrive (2025). Top 10 der verschiedenen Arten von Cyberangriffen — oodrive.com. <https://www.oodrive.com/de/blog/sicherheit/top-10-der-verschiedenen-arten-von-cyberangriffen/>. [Online; Stand 23.01.2025].
- [OpenSSL, 2025a] OpenSSL (2025a). EVP Symmetric Encryption and Decryption - OpenSSLWiki — wiki.openssl.org. https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption. [Online; Stand 16.04.2025].
- [OpenSSL, 2025b] OpenSSL (2025b). OpenSSL Documentation — docs.openssl.org. <https://docs.openssl.org/3.4/man1/openssl/#message-digest-commands>. [Online; Stand 28.04.2025].
- [Or, 2025] Or, J. B. (2025). Microsoft Corporation - Analyzing CVE-2025-31191: A macOS security-scoped bookmarks-based sandbox escape | Microsoft Security Blog — microsoft.com. <https://www.microsoft.com/en-us/security/blog/2025/05/01/analyzing-cve-2025-31191-a-macos-security-scoped-bookmarks-based-sandbox-escape/>. [Online; Stand 22.05.2025].
- [OWASP, 2025] OWASP (2025). Open Worldwide Application Security Project - Cryptographic Storage - OWASP Cheat Sheet Series — cheatsheetseries.owasp.org. https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html. [Online; Stand 21.04.2025].
- [Papathanasaki et al., 2022] Papathanasaki, M., Maglaras, L., and Ayres, N. (2022). Modern authentication methods: A comprehensive survey. *AI, Computer Science and Robotics Technology*.

- [Parecki, 2025] Parecki, A. (2025). OAuth 2.0 — OAuth — oauth.net. <https://oauth.net/2/>. [Online; Stand 15.05.2025].
- [Pohlmann, 2022] Pohlmann, N. (2022). Wirtschaftlichkeit von Cyber-Sicherheitsmaßnahmen.
- [Pohlmann, 2024] Pohlmann, N. (2024). Die Notwendigkeit von neuen IT-Sicherheitskonzepten. *Datenschutz und Datensicherheit-DuD*, 48(1):5–10.
- [Pufahl et al., 2024] Pufahl, M., Paulsen, P., and Arndt, P. (2024). Das Cybersecurity-Kompetenzmodell - Basis einer Ganzheitlichen Ausrichtung auf Cybergefahren.
- [Rimscha, 2018] Rimscha, M. (2018). Algorithmen.
- [Rominger et al., 2003] Rominger, A., Schmid, T., and Staats, M. (2003). Penetration Testing. In *Security, E-learning, E-Services, 17. DFN-Arbeitstagung über Kommunikationsnetze*, pages 715–734. Gesellschaft für Informatik eV.
- [Sänger, 2010] Sänger, S. (2010). Wer hohe Türme bauen will, muss lange beim Fundament verweilen. Zur Entwicklung des Deutschen Netzwerks Evidenzbasierte Medizin. *Zeitschrift für Evidenz, Fortbildung und Qualität im Gesundheitswesen*, 104(7):594–598.
- [Sangfor, 2025] Sangfor (2025). Sangfor Technologies - CVE-2025-2783: Google Chrome Sandbox Escape — sangfor.com. <https://www.sangfor.com/farsight-labs-threat-intelligence/cybersecurity/cve-2025-2783-google-chrome-sandbox-escape>. [Online; Stand 22.05.2025].
- [Schneider et al., 1999] Schneider, F., Council, N. R., et al. (1999). Committee on Information Systems Trustworthiness.(1999). *Trust in cyberspace*, page 214.
- [Schnieder and Schnieder, 2013] Schnieder, E. and Schnieder, L. (2013). Normativer Rahmen, Ziele und Organisationen.
- [Schünemann, 2023] Schünemann, W. J. (2023). Cybersicherheit. In *Handbuch Digitalisierung in Staat und Verwaltung*, pages 1–12. Springer.
- [Stringam and Gerdes, 2019] Stringam, B. and Gerdes, J. (2019). Service gap in hotel website load performance. *International Hospitality Review*, 33(1):16–29.

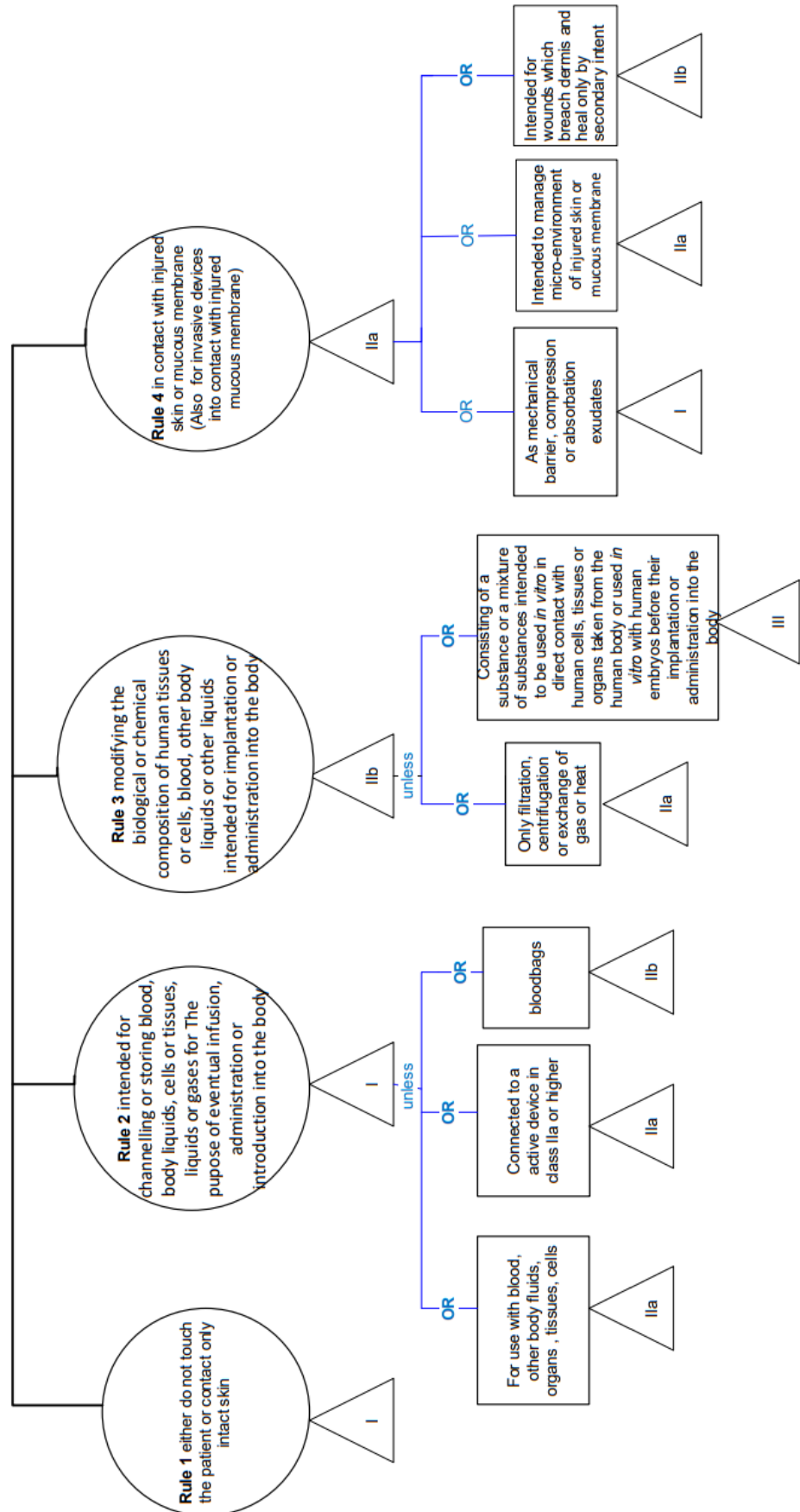
- [W3C, 2025] W3C (2025). World Wide Web Consortium - Web Authentication: An API for accessing Public Key Credentials - Level 3 — w3c.github.io. <https://w3c.github.io/webauthn/>. [Online; Stand 17.05.2025].
- [Wassel, 2025] Wassel, A. (2025). Software als Medizinprodukt: Definition und Klassifizierung — johner-institut.de. <https://www.johner-institut.de/blog/iec-62304-medizinische-software/software-als-medizinprodukt-definition/>. [Online; Stand 02.01.2025].
- [Wietzke and Tran, 2005] Wietzke, J. and Tran, M. T. (2005). Inter-Prozess-Kommunikationskanäle, IPC. *Automotive Embedded Systeme: Effizientes Framework—Vom Design zur Implementierung*, pages 121–150.
- [Yubico, 2025] Yubico (2025). GitHub - Yubico/Yubico.NET.SDK: A YubiKey SDK for .NET developers — github.com. <https://github.com/Yubico/Yubico.NET.SDK>. [Online; Stand 18.05.2025].
- [Zirkel, 2004] Zirkel, M. (2004). *Bewertung des UB-Baums unter Berücksichtigung der Sortierung*. PhD thesis, Technische Universität München.

Übersicht verwendeter Hilfsmittel

- DeepL SE Write: Zur Überprüfung von Rechtschreibung und Grammatik.
- Cornelsen Verlag GmbH Duden Mentor: Zur Überprüfung von Rechtschreibung und Grammatik.
- OpenAI ChatGPT4o: Hier wurden Ideen zur Struktur der Grundlagenkapitel gesammelt.

A. Anhang

A.1. Auszug aus der grafischen Zusammenfassung des MDCG-Dokuments 2021-24 zur Qualifizierung/Klassifizierung von Software als Medizinprodukt



A.2. Konkrete Anforderungen an Cybersicherheitsmaßnahmen nach Rechtsvorschrift für Software als Medizinprodukt

Anforderung	Herkunft
Personenbezogene Daten müssen in einer Weise verarbeitet werden, die eine angemessene Sicherheit der personenbezogenen Daten gewährleistet, einschließlich Schutz vor unbefugter oder unrechtmäßiger Verarbeitung und vor unbeabsichtigtem Verlust, unbeabsichtigter Zerstörung oder unbeabsichtigter Schädigung durch geeignete technische und organisatorische Maßnahmen. [EU, 2025b]	DSGVO
Es ist sicherzustellen, dass festgestellt werden kann, wer wann welche personenbezogenen Daten in welcher Weise verarbeitet hat. [EU, 2025b]	DSGVO
Der Hersteller muss Maßnahmen der Risikobeherrschung bestimmen, die geeignet sind, die Risiken auf einen akzeptablen Bereich zu mindern. [DIN, 2019]	ISO14971
Der Hersteller muss Schutzmaßnahmen im Medizinprodukt selbst oder im Herstellungsprozess bereitstellen. [DIN, 2019]	ISO14971
Der Zugang zu Informationen und anderen damit verbundenen Werten muss in Übereinstimmung mit der festgelegten themenspezifischen Richtlinie zur Zugangssteuerung eingeschränkt werden. [DIN, 2024]	ISO27001
Die Organisation muss die Anforderungen an die Wahrung der Privatsphäre und den Schutz personenbezogener Daten nach den geltenden Gesetzen und Vorschriften sowie den vertraglichen Anforderungen ermitteln und erfüllen. [DIN, 2024]	ISO27001
Maßnahmen zur Verhinderung von Datenlecks müssen auf Systeme, Netzwerke und alle anderen Geräte angewendet werden, die sensible Informationen verarbeiten, speichern oder übermitteln. [DIN, 2024]	ISO27001
Die Organisation muss Maßnahmen zur Beseitigung der Ursachen von möglichen Nichtkonformitäten ermitteln, um deren Auftreten zu verhindern. Vorbeugungsmaßnahmen müssen den Auswirkungen der möglichen Probleme entsprechen. [DIN, 2021]	ISO13485

A.3. Auszug der Bilddaten einer Datei

[illegible]

A.4. Detaillierte Testergebnisse der Verschlüsselungsmaßnahmen

Testfall	Durchschnittszeit (ms)	Maximalzeit (ms)	Durchschnittliche Zusatzzeit (ms)	Maximale Zusatzzeit (ms)
Landschaftsbild.bmp (75,8 MB)				
Unverschlüsselt Speichern	590,20	601,91	-	-
Unverschlüsselt Laden	329,31	335,21	-	-
XOR Speichern	708,88	711,00	118,68	381,69
XOR Laden	448,21	454,89	118,90	125,58
AES-256-CBC Speichern	909,36	948,74	319,16	619,43
AES-256-CBC Laden	641,88	646,74	312,57	317,43
Katzenbild.bmp (92,9 KB)				
Unverschlüsselt Speichern	4,11	4,59	-	-
Unverschlüsselt Laden	1,84	2,02	-	-
XOR Speichern	4,96	5,03	0,85	3,19
XOR Laden	2,41	2,44	0,57	0,60
AES-256-CBC Speichern	5,63	6,65	1,52	4,81
AES-256-CBC Laden	2,93	3,04	1,09	1,20

A.5. Detaillierte Testergebnisse der Kapselungsmaßnahmen

Testfall	Durchschnittszeit (ms)	Maximalzeit (ms)	Durchschnittliche Zusatzzeit (ms)	Maximale Zusatzzeit (ms)
Landschaftsbild.tif (101 MB)				
Einfache TIF-Laden	119,44	172,19	-	-
Einfache Kapselung	1355,08	1419,60	1235,64	1300,16
Verschlüsselte Kapselung	2199,23	2295,54	2079,79	2176,10
IPC Kapselung	786,08	872,60	666,64	753,16
Sandbox Kapselung	790,60	819,50	671,16	700,06
Katzenbild.tif (191 KB)				
Ohne Kapselung	2,77	3,01	-	-
Einfache Kapselung	56,83	64,33	54,06	61,56
Verschlüsselte Kapselung	60,54	68,29	57,77	65,52
IPC Kapselung	52,21	58,83	49,44	56,06
Sandbox Kapselung	53,98	60,72	51,21	57,95

Danksagung

Zunächst möchte ich mich bei der Image Instruments GmbH bedanken, welche mir die Möglichkeit gegeben hat, meine Masterarbeit im Rahmen eines betrieblichen Projekts umzusetzen. Ein besonderer Dank geht dabei an das Team, das mich mit seiner stetigen Betreuung, dem richtigen Maß an Kritik und Motivation entscheidend bei der Anfertigung der Masterarbeit unterstützt hat. Ein besonderes Dankeschön geht auch an die Professoren Thomas Franke und Rainer Wasinger, die diese Masterarbeit vonseiten der Hochschule ermöglicht haben. Zuletzt gilt mein besonderer Dank meiner Familie und meiner Freundin Anne Spreda, die mir nicht nur in dieser stressigen Zeit stets den Rücken freigehalten haben, sondern die auch ihre Zeit für das Korrekturlesen aufgebracht haben.



Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch keiner anderen Prüfung vorgelegen hat. Mir ist bewusst, dass ich Autor/in der vorliegenden Arbeit bin und volle Verantwortung für den Text trage.



Ich erkläre, dass ich wörtlich oder sinngemäß aus anderen Werken – dazu gehören auch Internetquellen – übernommene Inhalte als solche kenntlich gemacht und die entsprechenden Quellen angegeben habe.



Mir ist bewusst, dass meine Arbeit auf Plagiate überprüft werden kann. Mir ist bekannt, dass es sich bei der Abgabe eines Plagiats um ein schweres akademisches Fehlverhalten handelt und dass Täuschungen nach der für mich gültigen Prüfungsordnung geahndet werden.



Zusätzlich versichere ich, dass ich auf künstlicher Intelligenz (KI) basierende Werkzeuge nur in Absprache mit den Prüfern verwendet habe. Dabei stand meine eigene geistige Leistung im Vordergrund, und ich habe jederzeit den Prozess steuernd bearbeitet.



Diese Werkzeuge habe ich im Quellenverzeichnis in der Rubrik „Übersicht verwendeter Hilfsmittel“ mit ihrem Produktnamen und einer Übersicht des im Rahmen dieser Prüfungs-/Studienarbeit genutzten Funktionsumfangs unter Angabe der Textstelle in der Arbeit vollständig aufgeführt.



Ich versichere, dass ich keine KI-basierten Tools verwendet habe, deren Nutzung die Prüfer explizit schriftlich ausgeschlossen haben. Ich bin mir bewusst, dass die Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte Tools generiert wurden, keine Garantie für deren Qualität darstellt.



Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.



08056 Zwickau, 22.04.2025

Ort, Datum

Unterschrift