THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

ITI
Institut für Technik und Informatik

Bachelorarbeit

Transport sensibler Daten über nicht vertrauliche Netzwerke

unter Anwendung von C#

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt dem

Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen

Lars Kammerer

im November / 2022

Referent: Prof. Dr.-Ing. Andre Rein
Korreferent: Prof. Dr. Frank Kammer

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort, Datum Unterschrift

Gießen, 02.November.2022

Zusammenfassung

In der folgenden Arbeit wird eine Grundlage für eine Anwendung zum Transport sensibler Daten im Bereich der Praxisanwendungen, speziell Auswertegemeinschaften, erarbeitet. Die Sicherheit dieser Anwendung wird anhand der technischen Richtlinie des BSI [TR-03161] argumentiert und entwickelt. Der Fokus dieser Arbeit liegt auf der kryptografischen Umsetzung und der Anwendung kryptografischer Verfahren.

Dafür werden zunächst notwendige Grundlagen für kryptografische Verfahren und Anwendungen im Bereich der Medizin erläutert. In Kombination von beschriebenen Einschränkungen und der technischen Richtlinie des BSI wird daraufhin ein Katalog von Prüfaspekten zusammengestellt. Anhand dieser und der notwendigen Eigenschaften für eine Anwendung von Auswertegemeinschaften werden Anforderungen geformt. Die Umsetzung dieser Anforderung werden mit Fokus auf die kryptografische Umsetzung anhand eines Prototyps dargestellt. Die entstandenen Anforderungen und der Prototyp werden schlussendlich nochmals gegen den Katalog von Prüfaspekten gestellt, eventuelle Mängel festgestellt und Erfüllungen argumentiert. Die Arbeit endet mit einer Zusammenfassung und Aufarbeitung der Anwendung, die den Schluss zulässt, dass die erarbeitete Grundlage ausreichend sein kann.

Inhaltsverzeichnis

Bundesamt für Sicherheit in der Informationstechnik (BSI)	Amt für Beratung, Aufklärung und Richtlinienverfassung im Bereich IT-Sicherheit.[1]
Bundesinstitut für Arzneimittel und Medizinprodukte (BfArM)	Staatliches Institut für Arzneimittelzulassung, Aufklärung, Forschung und Informationsverbreitung im medizinischen Bereich. Zusätzlich zuständig für Bestandteile der eHealth-Infrastruktur bspw. der ePA.[2]
Datenschutz-Grundverordnung (DSGVO)	Gesetz zur Wahrung der Grundrechte von Personen innerhalb der EU, im Bezug auf deren Recht auf Schutz personenbezogener Daten, bei deren freien Verkehr und Verarbeitung.[3]
Digitale Gesundheitsanwendung (DiGA)	Meist mobile und patientennahe Anwendung zur Unterstützung bei Diagnostik, Genesung oder Linderung von Krankheitsbildern.[4]
Digitale Pflegeanwendung (DiPA)	Meist mobile und medizinische Anwendung zur Unterstützung von Pflegepersonal oder pflegebedürftigen Personen.[5]
Electronic Health (E-Health)	Oberbegriff für digitale Anwendungen im medizinischen Sektor. Bspw. ePA, DiPA, DiGA, E-Rezepte usw.[6]
Elektronischen Patientenakte (ePA)	Digitale Form einer Patientenakte. Kann Diagnosen, Untersuchungen, Rezepte o.ä. beinhalten.[7]
Langzeit-EKG	Langfristig angelegte Messung der Herzaktionen (EKG).
Mobile Health (mHealth)	Untergruppe von E-Health. Beschränkt sich auf mobile Anwendungen.[8]
Patientendaten-Schutz-Gesetz (PDSG)	Gesetzliche Regelung zum Schutz von Patientendaten, deren Nutzung und Verarbeitung.[9]
Klartext	Unverschlüsselte Information oder Datei. Kann das Ergebnis einer Entschlüsselung sein.
Ciphertext/Chiffrat	Verschlüsselte/Chiffrierte Information oder Datei als Ergebnis einer Verschlüsselung.
Certificate Authority (CA)	Instanz zur Ausstellung, Prüfung und Aufbewahrung von Zertifikaten[10, 11, 12]

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Im Verlaufe der Jahre entwickelte sich die Digitalisierung im deutschen Gesundheitswesen rasant. Änderungen im Patientendaten-Schutz-Gesetz (PDSG) ermöglichen immer weitere digitale Lösungen für medizinische Einrichtungen. E-Health ist ein stetig wachsender Markt. Medikationsplan, Befunde, Notfallkontakte oder aktuelle Behandlungen, all das findet sich nach und nach in der ePA der elektronischen Patientenakte wieder. In Krankenhäusern nimmt die "Zettelwirtschaft" immer weiter ab. Informationen zu jeder Zeit, überall im Krankenhaus, abrufen zu können bietet einen großen Vorteil gegenüber der Akte aus Papier. [14, 15]

E-Health bietet einen großen Vorteil für Patienten und medizinische Einrichtungen.[15] Doch die Skepsis von vorwiegend älteren Patienten besteht weiterhin. 2019 nutzen 93% der Ärzte den analogen Kommunikationsweg mit Krankenhäusern und das, obwohl mehr als acht von zehn Ärzten an Telematikinfrastruktur angebunden sind. Knapp 60% der Praxen im PraxisBarometer der KBV haben keinerlei digitalen Service im Sortiment. Laut "eHealth Monitor 2020" äußerten sich 43% der Ärzte und Apotheken skeptisch gegenüber Digitalisierung. Grund sei eine vermutete Verschlechterung der Arzt-Patienten-Beziehung.[16]

Aus diesen Zahlen lassen sich entsprechend ähnliche Werte für Auswertegemeinschaften ableiten. Auswertegemeinschaften bestehen aus mehreren Allgemeinmedizinern, hier weiter als Zuweiser, und mindestens einem Facharzt. Es sind meist kleine Zusammenschlüsse auf ländlicher Ebene die, um Kosten und Aufwand zu sparen, sich in einer Auswertegemeinschaft zusammenfinden. Allgemeinmediziner sind, auf den Anwendungsfall eines Langzeit-EKG betrachtet, diejenigen, die eine entsprechende Aufzeichnung im Falle einer Nachuntersuchung oder eines Verdachtsfalls vornehmen. Diese Aufzeichnung muss jedoch durch einen Facharzt, hier weiter als Auswerter, ausgewertet werden. Eine Auswertegemeinschaft definiert sich also dadurch, dass mehrere Zuweiser ihre Aufzeichnungen durch einen Auswerter durchführen lassen. Ein Auswerter kann auch ein Zusammenschluss mehrerer Fachärzte sein.

Doch die Corona-Pandemie verändert diese Zahlen deutlich. Im Jahr 2020 boten 52% der Arztpraxen Videosprechstunden an, im Vergleich dazu waren es im Jahr 2017 nur 2%. Die digitalen Angebote finden Anklang. So begrüßen laut Umfragen zwei von drei Deutschen die Einführung der ePA und eRezepte. Bei den über 65-Jährigen sind es 60%. 2018 wurde Deutschland noch durch die Bertelsmann Stiftung bei dem Indikator *"mHealth, Apps und mobile Anwendungen werden routinemäßig in der Gesundheitsversorgung genutzt"* mit der niedrigsten Stufe *"nicht vorhanden"* bewertet. 2020 sehen diese Werte laut dem "eHealth Monitor 2020" besser aus.[16]

Digitaler Fortschritt soll jedoch überall ankommen, damit auch kleine Praxen die Vorteile einer schnellen und zuverlässigen Kommunikation für sich nutzen können. Also auch Auswertegemeinschaften. Die Probleme dieser Auswertegemeinschaften finden sich in der Kommunikation. Ist eine Aufzeichnung durch einen Zuweiser angefertigt, so sind diese in digitaler Form auf einer Speicherkarte abgelegt. Die Übertragung dieser Daten fand in der Vergangenheit über beispielsweise den Postweg statt. Eine entsprechende Übertragung kostet also Zeit und die versendete Karte kann, sofern die Daten nicht vorher auf ein anderes Medium übertragen wurden, nicht erneut genutzt werden. Zumindest solange sie sich unterwegs befindet.

Lösungen für die Kommunikation zwischen Zuweiser und Auswerter bieten bereits einige Unternehmen an. Auf einigen Webseiten dieser Unternehmen sind jedoch keine Zertifikate zu finden, die auf eine Prüfung des BSI hinweisen. Auch fehlen sie auf der Liste der geprüften Unternehmen des BSI.[17] Dies lässt die Vermutung zu, dass keine Sicherheitsprüfung durch das BSI durchgeführt wurde. Das Interesse, eine nach den BSI Richtlinien geprüften Lösung anzubieten, ist daher ein logischer Schritt in die Digitalisierung von Auswertegemeinschaften.

1.2 Problembeschreibung

Im Jahr 2021 hat das BSI in der Statistik *"IT-Sicherheit auf dem digitalen Verbrauchermarkt: Fokus Gesundheits-Apps"* offengelegt, dass einige Anbieter von Anwendungen im Gesundheitswesen Verbesserungsbedarf aufweisen.[18] Für Patienten als auch für Praxen ist es in der voranschreitenden Digitalisierung daher wichtig, sich auf entsprechende Anwendungen verlassen zu können, vor allem in Puncto Sicherheit und Datenschutz. Das Patientendaten-Schutz-Gesetz und die Datenschutz-Grundverordnung (DSGVO) stellen klar, dass personenbezogene Daten und besonders Patientendaten unter besonderem Schutz stehen.[14, 3] Anwendungen die mit Patientendaten arbeiten, unterliegen deshalb bestimmten Vorschriften und Vorgaben, die in jedem Falle eingehalten werden müssen, sofern die Anwendung durch eine Behörde geprüft werden soll. In Deutschland existieren mehrere Prüforganisationen. Welche Prüforganisation für welche Anwendung zuständig ist, hängt alleine von dem Zweck der Anwendung ab. Um Vertrauen in digitale Anwendungen im Gesundheitswesen zu schaffen, ist es also von hohem Interesse, Lösungen mit einem hohen und geprüften Sicherheitsstandard vorweisen zu können. Die Problematik besteht demnach darin, Anwendungen anhand dieser Standards zu entwickeln und entsprechende Prüfungen zu bestehen.

Im Folgenden soll die Zuweiser-Auswerter-Kommunikation mit Hilfe einer Softwarelösung sicherer gemacht werden. Bei der Zuweiser-Auswerter-Kommunikation oder auch Auswerter-Zuweiser-Kommunikation handelt es sich um die Kommunikation zwischen zwei Parteien einer Auswertegemeinschaft. In einer Auswertegemeinschaft existieren zwei Parteien: Zuweiser und Auswerter. Zuweiser übergeben Aufträge für eine Auswertung an einen Auswerter. Dieser fertigt eine Auswertung an und übergibt sie zurück an den entsprechenden Zuweiser. Dieses Konzept findet im medizinischen Bereich z.B. zwischen Allgemeinmedizinern und einem Facharzt wie einem Kardiologen eine Anwendung. Hierbei ist der Allgemeinmediziner der Zuweiser und der Kardiologe der entsprechende Auswerter. Dadurch können Patienten, ohne einen Facharzt zu konsultieren, bei ihrem Hausarzt dennoch eine Fachuntersuchung in einigen Gebieten erhalten.

Die Herausforderung besteht demzufolge darin eine Anwendung für diese Zuweiser-Auswerter-Kommunikation zu entwickeln, die die Vorgaben der entsprechenden Prüforganisation abdeckt und einer eventuellen Prüfung standhält. Gleichzeitig sollen Lösungen für den speziellen Anwendungsfall dieser Kommunikation optimiert sein. Der Fokus einer solchen Anwendung liegt, aufgrund der sensiblen Daten, auf der Sicherheit und dem Datenschutz. Entsprechend liegt das Kernproblem in einem sicheren Kommunikationssystem und dessen Infrastruktur.

1.3 Zielsetzung und Abgrenzung

Ziel ist es eine Client-seitige Anwendung zu schaffen, die eine Grundlage für die Lösung der Zuweiser-Auswerter-Kommunikation bietet. Die Eigenschaften der Anwendung sind auf Sicherheit und Korrektheit abzugleichen, um so möglichst nahe an eine optimale Grundlage für eine Lösung zu gelangen. Dabei soll die Grundlage als Prototyp einer Anwendung dargestellt werden. Um Korrektheit und Sicherheit prüfen zu können, müssen sich Anforderungen, Design und Implementierung an den technischen Richtlinien der entsprechenden Prüforganisation orientieren. Diese soll Leitfaden und Prüfmittel für die Eigenschaften des Prototyps darstellen.

Das Konzept soll innerhalb eines geringen Zeitfensters implementierbar sein und für eine Weiterentwicklungen bereit sein. Daher sind Einschränkungen durch Implementierungsaufwand zu berücksichtigen. Durch die Betrachtung einer Prototypentwicklung entfallen Prüfaspekte, die einen solchen Status überschreiten. Bspw. entfallen Prüfaspekte, die sich mit sicherheitspolitischen Entscheidungen beschäftigen. Der Fokus der betrachteten Aspekte liegt klar auf der kryptografischen Umsetzung. Dennoch werden Themen wie die Handhabe der DSGVO behandelt, um das Gesamtbild im Umgang mit sensiblen Daten anzureißen. Auch wenn diese Thematik nicht weniger wichtig ist, wird keine detaillierte Betrachtung dieser stattfinden. Zusätzlich wird die Nutzung von Hardware-Sicherheitsmaßnahmen nicht berücksichtigt.

Die Hardware bzw. das Endgerät der Anwender ist ein Desktop-Rechner/Stand-Rechner, dessen Betriebssystem mindestens eine aktuelle Windows 10 Version ist. Da sich die Prüfung auf eine Desktop-Anwendung bezieht entfallen dadurch bspw. Prüfaspekte mit Bezug auf mobile Endgeräte. Es wird davon ausgegangen, dass eine Installation auf den Systemen des Clients grundsätzlich reibungslos funktioniert. Eine Manipulation der Software während der Installation ist von vornherein ausgeschlossen. Das gilt auch für bei der Installation eventuell mitgelieferte Zertifikate. Zudem werden Rückmeldungen des Servers in einer gewissen Form erwartet. Entspricht die Antwort des Servers nicht dieser Form, wird sie grundsätzlich abgelehnt bzw. verworfen. Dies schützt den Client jedoch nicht vor manipulierten Daten, sondern lediglich vor eventuellen Schwachstellen in der Paketübernahme.

Durch eine unabhängige Quelle ausgestellte Zertifikate sind grundsätzlich als vertrauenswürdig eingestuft. Eventuelle Schwachstellen einer Certificate-Authority (CA) stehen außer Frage. Der Einfachheit halber wird die Client-Server-Kommunikation vereinfacht betrachtet, Zwischenschritte wie ISPs werden außer Acht gelassen. Der Anbieter des Kommunikationskanals ist zu jederzeit als nicht vertrauenswürdig einzustufen. Ebenfalls ist das Netzwerk des Clients nicht als vertrauenswürdig einzustufen. Damit sind Manipulationen innerhalb dieser Netze eine Bedrohung. Der Verbindungsaufbau eines Clients zum Server, bspw. über TLS, wird immer als reibungslos und sicher betrachtet. Das Hintergrundsystem (Server) wird nicht im Detail betrachtet oder geprüft, die Anforderungen an das Hintergrundsystem ergeben sich aus den Eigenschaften der Anwendung.

Alle Lösungsansätze sind stets unter Berücksichtigung der, in der Programmiersprache C# (.NET-Version 6), zur Verfügung gestellten Mittel zu entwickeln. Drittanbieter-Pakete sind von diesen Mitteln ausgeschlossen. Eine Begründung für diese Entscheidung wird dargelegt.

1.4 Vorgehensweise

Um einen geeigneten Katalog an Prüfaspekten zu erarbeiten, muss zu Beginn festgestellt werden, welche der, in Deutschland bekannten Prüforganisationen, für eine Anwendung mit entsprechenden Eigenschaften zuständig ist. Ist diese gefunden müssen alle nötigen Richtlinien geprüft und festgestellt werden, welche dieser Richtlinien am besten auf dieses Anwendungsgebiet passen. Infolgedessen ist festzustellen, wie eine Prüfung stattfindet und welche Testcharakteristika/Prüfaspekte angewendet werden. Um einen Leitfaden für die Implementierung zu schaffen, werden zunächst die in der Richtlinie vorhandenen Prüfaspekte aufgegriffen und erläutert. Zusätzlich werden diese Aspekte anhand der Zielsetzung und Abgrenzung eingeschränkt und gefiltert. Unter Berücksichtigung der Eigenschaften einer Auswertegemeinschaft und der eingeschränkten Prüfaspekte ergeben sich Anforderungen an die Implementierung. Darauf folgt die Erläuterung des auf diesen Anforderungen basierenden Prototypen. Dieser bezieht sich dabei auf die kryptografische Umsetzung und die Anwendung von entsprechenden Verfahren und dessen Zusammenarbeit. Die Anforderungen und der Prototyp ergeben zusammen die Grundlage für die Lösung der Auswerter-Zuweiser-Kommunikation. Diese Grundlage wird im Anschluss gegen die eingeschränkten Prüfaspekte gestellt und eventuelle Verfehlungen wie auch Erfüllungen werden argumentiert. Im Abschnitt "Alternativen" finden nicht umgesetzte alternative Vorgehensweisen einen Platz. Die Arbeit endet mit einer Zusammenfassung und Aufarbeitung der erarbeiteten Grundlage.

2 Grundlagen

2.1 Auswertegemeinschaften

Auswertegemeinschaften sind eine oder mehrere Praxen, meist Kardiologen, die Verbindungen zu Praxen der Allgemeinmedizin pflegen. Diese Verbindung besteht durch die Notwendigkeit, dass Allgemeinmediziner nicht das nötige Know-how oder Material besitzen, um eine fachärztliche Aufzeichnung auszuwerten. Durch eine Auswertegemeinschaft sind Allgemeinmediziner in der Lage, solche Aufzeichnungen auswerten zu lassen. Meist befinden sich mehrere Allgemeinmediziner in einer Auswertegemeinschaft mit nur einem Auswerter der entsprechenden fachärztlichen Aufzeichnungen auswerten kann. Für den Auswerter bedeutet dies weniger Untersuchungsaufwand für Patienten, die auch bspw. durch Hausärzte untersucht werden können. Zwar übernimmt der Auswerter weiterhin die Auswertung der Untersuchung, aber der Patient selbst bleibt weiterhin der Patient des Hausarztes. Das spart dem Auswerter Arbeit und Kapazität. Dadurch ist er in der Lage mehr Patienten als üblich auszuwerten, was wiederum den Umsatz steigert.

2.2 GDT-Dateien

Geräte-Datenträger-Datei kurz GDT-Datei ist eine Schnittstellenbeschreibung zum Transport von Daten (speziell medizinische Daten) zwischen Arzt-Informationssystemen und medizinischen Messgeräten. Die Datei enthält mehrere Felder, die unter anderem Patientendaten beinhalten. Jedes Feld folgt dem gleichen Aufbau. Ein Feld besteht aus vier Komponenten: Zeilenlänge, Feldkennung, Inhalt und Zeilenende.

Bsp. für einen Eintrag für den Nachnamen:
0193101Mustermann[CR][LF]

Die ersten drei Ziffern beschreiben die Zeilenlänge. Sie ergibt sich aus ihrer eigenen Länge ([019]→Länge drei), der Länge der Feldkennung ([3101]→Länge vier), dem Inhalt ([Mustermann]→Länge zehn) und dem Zeilenende ([CR][LF]→Länge zwei). Länge insgesamt, 19. Bis auf den Inhalt haben alle Komponenten immer die gleiche Länge. Die Feldkennung gibt vor, was sich im Feld befindet. Bspw.: Vorname (3102) oder Diagnose (3622). Der Inhalt ist die eigentlich zu übermittelnde Information und [CR][LF] ist das Ende der Zeile.

2.3 Personenbezogene Daten

Als personenbezogene Daten werden all die Daten bezeichnet, die einen klaren Rückschluss auf eine lebende Person zulassen und/oder diese ausmachen oder beschreiben. Dazu zählen eindeutige Fälle, wie bspw. der Nachname. Daten die in Kombination einen Rückschluss zulassen, Fallen auch in diese Kategorie, zum Beispiel GPS-Daten oder eine Anamnese. [3, 19]

2.4 Kryptografische Grundlagen

2.4.1 Symmetrische Verschlüsselung

Das Verfahren der symmetrischen Verschlüsselung basiert auf dem Grundprinzip, dass ein geheimer Schlüssel die Vertraulichkeit von Daten sichert. Die vertraulichen Daten werden dabei mit dem Schlüssel chiffriert und können entsprechend nur von Besitzern des Schlüssels entschlüsselt werden. Haben sich bspw. Sender und Empfänger auf einen geheimen Schlüssel geeinigt, kann dieser genutzt werden, um geheim bzw. vertraulich Nachrichten auszutauschen. So soll es Dritten nicht möglich sein, diese Nachrichten, aufgrund mangelnder Kenntnisse über den Schlüssel, zu Entschlüsseln und so den Klartext zu lesen. Die Sicherheit von modernen Verschlüsselungen soll allein auf der Sicherheit/Stärke des Schlüssels basieren. Sprich, das Offenlegen des Algorithmus gefährdet keine bestehende Verschlüsselung. Diese Algorithmen greifen auf einfache Operationen zurück, bspw. XOR, AND oder OR. Dies macht sie besonders effizient, vor allem für einen größeren Verschlüsselungsbedarf.[12, 20, 21]

Definition:
Sei $Encyption_{Sym}(Key_{Sym}, K)$ eine Methode für eine symmetrische Verschlüsselung, so bedarf diese einen symmetrischen Schlüssel Key_{Sym} und einen zu verschlüsselnden Klartext K, um einen Ciphertext $Cipher$ zu erzeugen. Das Gegenstück zur Verschlüsselung bietet die Entschlüsselungsmethode $Decyption_{Sym}(Key_{Sym}, Cipher)$.

Daraus folgt:
$Encyption_{Sym}(Key_{Sym}, K) \Rightarrow Cipher$
$Decyption_{Sym}(Key_{Sym}, Cipher) \Rightarrow K$

2.4.2 Asymmetrische Verschlüsselung

Asymmetrische Verschlüsselung basiert auf der Existenz sogenannter Schlüsselpaare. Jedes Schlüsselpaar besteht dabei aus einem öffentlichen und einem privaten Schlüssel. Wie die Namen bereits implizieren, ist der private Schlüssel stets geheim zu halten, während der öffentliche Schlüssel öffentlich (für jeden zugänglich) bekannt sein darf. Das Prinzip dahinter basiert auf der mathematischen Möglichkeit eine Nachricht mit dem öffentlichen Schlüssel zu chiffrieren und infolgedessen **nur** mit dem, zu diesem Schlüssel passenden, privaten Schlüssel zu entschlüsseln. Sprich: Möchte ein Sender eine Nachricht verschlüsselt übertragen, nutzt er bei diesem Verfahren den öffentlichen Schlüssel des Empfängers, um diese zu chiffrieren. Ausschließlich der Empfänger kann nun die Nachricht mit seinem privaten Schlüssel entschlüsseln, sonst niemand. Dieses Verfahren kann jedoch auch umgekehrt angewendet werden. Eine Nachricht verschlüsselt mit dem privaten Schlüssel kann mit dem öffentlichen Schlüssel entschlüsselt werden. Ein Verfahren, welches bei digitalen Signaturen zum Tragen kommt. Asymmetrische Verfahren sind jedoch erheblich aufwendiger als symmetrische Verfahren. Zusätzlich ist die Länge der zu chiffrierenden Nachricht begrenzt.[12, 20, 21, 22]

Die mathematische Grundlage für diese Verfahren bieten die Probleme der Faktorisierung (bspw. RSA), und des Diskreter-Logarithmus (DL) sowie das Diffie-Hellman-Problem (DH-Problem). Jedes Problem bedarf eine anderen Schlüsselerzeugung und Herangehensweise an Ver- und Entschlüsselung. Daher sind bspw. RSA-Schlüssel nicht mit Schlüsseln basierend auf dem DL-Problem "kompatibel".[12, 20, 21, 22]

Definition nach RSA:
Es existiert ein Schlüsselpaar $KeyPair$ eines Empfängers, bestehend aus dem öffentlichen Schlüssel Key_{Pub} und einem privaten Schlüssel Key_{Priv}.

Ist $Encryption_{Asym}(Key_{Pub}, K)$ eine Funktion für eine asymmetrische Verschlüsselungsmethode, so benötigt diese den öffentlichen Schlüssel des Empfängers Key_{Pub}, um einen zu chiffrierenden Klartext K zu einem Ciphertext $Cipher$ zu verschlüsseln. Die Entschlüsselung kann mit einer Methode $Decryption_{Asym}(Key_{Priv}, Cipher)$ erfolgen. Diese benötigt entsprechend den Ciphertext $Cipher$ und den privaten Schlüssel des Empfängers Key_{Priv}.

Daraus folgt:
$Encryption_{Asym}(Key_{Pub}, K) \Rightarrow Cipher$
$Decryption_{Asym}(Key_{Priv}, Cipher) \Rightarrow K$

2.4.3 Hybride Verschlüsselung

Unter hybriden Verschlüsselungsverfahren versteht sich das Zusammenspiel zwischen einer asymmetrischen und symmetrischen Verschlüsselung. Eine symmetrische Verschlüsselung hat die Eigenschaft, dass jeder, der den zugehörigen Schlüssel besitzt, die chiffrierten Nachrichten entschlüsseln kann. Das bedeutet der Sender und Empfänger müssen sich auf einen gemeinsamen Schlüssel einigen, ohne dass dieser in falsche Hände gerät. Dies erzeugt ein signifikantes Problem - den Schlüsselaustausch. Um diesen möglich zu machen, können asymmetrische Verfahren zum Einsatz kommen. Bspw. kann der symmetrische Schlüssel mit dem öffentlichen Schlüssel des Empfängers chiffriert und so unbrauchbar für unbeteiligte Dritte gemacht werden. Durch die Eigenschaften der asymmetrischen Verschlüsselung kann lediglich der Empfänger bzw. jene Person, die den passenden privaten Schlüssels besitzt, diesen symmetrischen Schlüssel entschlüsseln und für einen vertraulichen Nachrichtenaustausch mit dem Sender verwenden.[20, p. 173-178]

Key Transport Schemes/Schlüssel-Transport-Verfahren sind Verfahren, die unter Anwendung von hybrider Verschlüsselungen einen symmetrischen Schlüssel vertraulich an einen Kommunikationspartner übertragen. Die Übertragung des symmetrischen Schlüssels wird durch Chiffrierung mit dem öffentlichen Schlüssel des Empfängers ermöglicht. Bsp. für solche Verfahren sind ECIES und DLIES.[23][22, p. 107]

2.4.4 Message Authentication Code (MAC)

"Message Authentication Codes" (MACs) generieren sich aus der Nachricht und einem geteilten und geheimen symmetrischen Schlüssel. Sie sollen Integrität und Authentizität einer Nachricht sicherstellen. Die Prüfbarkeit von Integrität wird bspw. mit "Secure Hash Algorithms" (SHA2) hergestellt (auch Message Integrity Code [MIC]). Unter zusätzlicher Anwendung des geteilten symmetrischen Schlüssels zwischen Sender und Empfänger kann die Integritätsprüfung um eine Authentizitätsprüfung erweitert werden. Das bedeutet, durch die Kombination wird sichergestellt, dass die Nachricht nicht verändert wurde (Integrität) und der "Message Authentication Code" ausschließlich von einem Teilnehmer mit Zugang zum geteilten Schlüssel generiert wurde (Authentizität).[12, p. 377][24]

2.4.5 Digitale Signaturen (DS)

Zur Erzeugung von **"Digital Signatures" (DS)** *(DE: digitale Signaturen)* kommen Hash-Algorithmen und asymmetrische Verfahren zum Einsatz. Zunächst muss von der Nachricht, welche später durch einen Empfänger verifiziert werden soll, ein Hashwert gebildet werden. Der erzeugte Hashwert wird dann unter Verwendung des privaten Schlüssels des Senders signiert. Die Eigenschaften von asymmetrischen Verfahren machen es möglich, dass nun jeder, der im Besitzt des passenden öffentlichen Schlüssels ist, die Signatur verifizieren kann.[21, p. 209-301]

Digitale Signaturen besitzen, wie MACs, die Eigenschaft, Integrität nachweisen zu können. Zusätzlich können sie jedoch auch Nachweisbarkeit bzw. eine nichtabstreitbare Urheberschaft nachweisen.[25] Das bedeutet: Die Signatur, und damit auch die Nachricht, wurde durch den erwarteten Sender erzeugt bzw. der Sender ist im Besitz des erwarteten privaten Schlüssels. Solange der Sender seinen privaten Schlüssel geheim hält, ist es unbefugten Dritten, nicht möglich, Signaturen zu fälschen.

Anmerkung: Im Verlauf dieser Arbeit werden die Begriffe digitale Signatur und Signatur äquivalent gebraucht.

2.4.6 Zertifikate

Zertifikate sind eine Möglichkeit, asymmetrische öffentliche Schlüssel verifizieren zu können. Dazu enthält das Zertifikat, neben zusätzlichen Informationen zur Identität des Besitzers, den jeweiligen Schlüssel und eine Signatur. Diese Signatur wird durch einen privaten Schlüssel erzeugt und soll mit Hilfe eines bekannten öffentliche Schlüssels (meist ebenfalls Zertifikate) die Integrität und Nachweisbarkeit des Zertifikats prüfbar machen. Eine Möglichkeit, vertrauenswürdige Zertifikate zu erzeugen und zu verifizieren, ist bspw. eine CA.[10, p. 137-138]

2.4.7 Mutual authetifikation

Bei "Mutual authetifikation" (gegenseitiger Authentifizierung) wird meist, anstelle eines Passworts, auf die Sicherheit eines asymmetrischen Verfahrens und dessen kryptografischen Eigenschaften gesetzt. Sprich: Der Server muss den öffentlichen Schlüssel des Nutzers kennen und mit diesem nachweisen, dass der zu authentifizierende Nutzer im Besitzt des dazu gehörigen privaten Schlüssels ist. Dies geschieht z.B. im Fall von SSH-1 durch das Übergeben einer, mit dem öffentlichen Schlüssel des Clients verschlüsselten, zufälligen Zeichenkette durch den Server an den Client. Dies wird als Challange/Issue bezeichnet. Der Client entschlüsselt mit seinem privaten Schlüssel diese Zeichenkette, erzeugt aus dieser einen Hash und übergibt diesen zurück an den Server. Kann der Server den Hashwert reproduzieren, ist klar: Der Client besitzt einen privaten Schlüssel zu dem bekannten öffentlichen Schlüssel und die Authentifizierung ist erfolgreich.[26, 27, 11] Im Gegensatz zu einer "klassichen" Authentifizierung mit einem Passwort besteht bei einem kryptografisch basierenden Verfahren kein Risiko durch das Erraten von dieser und zusätzlich erspart es dem Anwender entsprechende Passwörter.[28]

3 Prüforganisationen

Prüforganisationen sind Organisationen, meist Behörden, die eine Einhaltung von Standards von Anwendungen, Geräten, Gebäuden usw. zertifizieren.

Für Anwendungen im Gesundheitswesen existieren in Deutschland drei potenzielle Prüforganisationen: der Technischer Überwachungsverein (TÜV), das Bundesinstitut für Arzneimittel und Medizinprodukte (BfArM) und das Bundesamt für Sicherheit in der Informationstechnik (BSI). Welche dieser drei Organisationen als Prüfer infrage kommt, hängt allein vom Zweck bzw. vom Einsatzbereich der Anwendung ab. Der Einsatzbereich wird in zwei primäre Kategorien unterteilt:

1. Medizinprodukte
2. Praxisausstattung

Ein medizinisches Produkt bzw. ein Medizinprodukt ist ein Produkt, welches eine patientennahe Anwendung hat, bspw. um Diagnosen zu ermöglichen oder einfache Überwachungstätigkeiten zu übernehmen.[29, 4, 30, 31] In diese Kategorie fallen unter anderem *"Digitale Gesundheitsanwendungen (DiGA)"* und *"Digitale Pflegeanwendungen (DiPA)"*. DiGA und DiPA sind patientennahe Anwendungen die Leben und/oder Genesung dieser Patienten verbessern bzw. unterstützen sollen. Bspw. existieren DiGA die Patienten mit Burnout, Depression oder chronische Schmerzen begleiten sollen und diese z.B. mit regelmäßigen Kursen versorgen. Entsprechende Anwendungen begrenzen sich meist auf Webanwendungen oder Apps.[5] Ihre Kontrolle obliegt dem BfArM und dem TÜV. Welche DiGA/DiPA geprüft ist, kann aus dem vom BfArM zur Verfügung gestellten Verzeichnis entnommen werden.[32] In die Kategorie *Praxisausstattung* fallen Produkte, welche in der Medizin Anwendung finden, jedoch nicht direkt mit dem Patienten zu tun haben. Also nicht patientennah sind. Solche Anwendungen nennt man *allgemeine Anwendung im Gesundheitswesen* oder *Praxisausstattung*.

Die in dieser Arbeit betrachtete Anwendung soll einen sicheren Kommunikationskanal zwischen Praxen herstellen. Eine solche Anwendung fällt laut Definition in den Bereich der *allgemeine Anwendung im Gesundheitswesen* bzw. in die Kategorie *Praxisausstattung*, da diese eben nicht patientennah eingesetzt wird. Software, die als Praxisausstattung zählt, fällt in die Zuständigkeit des BSI, da weder TÜV noch BfArM hier zuständig sind. Es gelten entsprechend auch die von dieser Behörde vorgegebenen Richtlinien. Entsprechende Vorgaben sind durch alle genannten Behörden für jeden zugänglich.

Die passende technische Richtlinie TR-03161 *Anforderungen an Anwendungen im Gesundheitswesen*[13] teilt sich in drei Teile. Teil eins für mobile Anwendungen, zwei für Web-Anwendungen und drei für Hintergrundsysteme. Da sich die hier behandelte Software auf den Einsatz auf Windows-Desktop Systemen beschränkt passt keine der genannten Teile. Zwar nutzt die Software ein Hintergrundsystem auf, das der dritte Teil der TR passen würde, jedoch ist diese nur bedingt nützlich für die in dieser Arbeit beschrieben Anwendung. Das BSI schreibt zu diesen technischen Richtlinien:

> *"Die Familie von Technische Richtlinien (TR) richtet sich an Hersteller von Anwendungen im Gesundheitswesen. Zusätzlich kann sie als Richtlinie für Anwendungen betrachtet werden, welche sensible Daten verarbeiten oder speichern."*
> *- BSI TR-03161 Anforderungen an Anwendungen im Gesundheitswesen*[13]

Es lässt sich entnehmen, dass zwar kein spezifischer Teil der TR existiert, der auf die Anwendung zutrifft, jedoch ist es möglich, die Richtlinien als Leitfaden zu nutzen. Betrachtet man den Prozess der Prüfung wird auch ersichtlich, dass nicht alle Prüfaspekte angewendet werden müssen.[13, p. 27] Vor dem eigentlichen Prüfverfahren wird in Zusammenarbeit mit dem Hersteller der Anwendung ein Katalog von Testcharakteristika zusammengestellt, der für die Prüfung notwendig ist. Hierbei hat der Hersteller nur bedingt Einfluss auf die gewählten Charakteristika. Er kann jedoch unterstützend auf den Prozess einwirken, indem er bspw. die Umsetzung genau skizziert. Die Testcharakteristika leiten sich aus den Prüfaspekten ab, welche wiederum als Leitfaden für die Implementierung der Anwendung dienen. Für die Prüfung der hier beschriebenen Anwendung wird Teil eins der TR genutzt. Dieser trifft am ehesten auf eine Windows-Desktop-Anwendung zu und enthält alle notwendigen Prüfaspekte für eine Anwendung auf einem Endgerät, als auch einige zusätzliche für die Nutzung eines mobilen Endgeräts.

Für die in dieser Arbeit beschriebene Anwendung gelten entsprechend die TR-03161-1[13] in Kombination mit der TR-02102-1 ink. -2[33] und TR-02103[34]

4 Prüfaspekte

Im folgenden Kapitel werden die, für den im Verlauf vorgestellten Prototyp, angedachten Prüfaspekte des BSI angegeben und gefiltert. Die Prüfaspekte sind in ihrem vollen Umfang der TR-03161-1[13] zu entnehmen. Es ist wichtig zu verstehen, dass hier nur die Sicht des Herstellers wiedergegeben wird. Eine vollumfängliche, reale Überprüfung der Anwendung kann und wird in dieser Arbeit nicht stattfinden können. Für die Auswahl der Prüfaspekte gelten die im Kapitel *1.3 Zielsetzung und Abgrenzung* beschriebenen Grundsätze. Es entfallen daher Prüfaspekte für Drittanbieter-Pakete und mobile Anwendungen. Bedingt werden auch Prüfaspekte in Bezug auf das Hintergrundsystem und die damit zusammenhängende Authentifizierung großzügig ausgelassen, da der Fokus auf der Client-Anwendung liegt. Es ist zusätzlich zu bedenken, dass es sich bei dieser Betrachtung nur um einen Prototyp handelt. Der Fokus der Prüfaspekte soll auf den Umgang mit sensiblen Daten, deren sicheren Transport und den damit zusammenhängenden angewendeten kryptografischen Verfahren liegen.

Im Gesamten unterteilen sich die Prüfaspekte in elf Kategorien. Jede dieser Kategorien besitzt mehrere Prüfaspekte. Diese sind durch einen Bezeichner und eine Nummerierung unterscheidbar.

Nr.	**Kategorie**	**Bezeichner (x = Platzhalter Nummerierung)**
1	Anwendungszweck	O.Purp_x
2	Architektur	O.Arch_x
3	Quellcodes	O.Source_x
4	Drittanbieter-Software (third party)	O.TrdP_x
5	Kryptographischen Umsetzung	O.Cryp_x und O.Rand_x
6	Authentifizierung: Passwörter, Biometrischedaten, Sitzungen und Tokens	O.Auth_x O.Pass_x, O.Biom_x, O.Sess_x und O.Tokn_x
7	Datenspeicherung u. Datenschutz	O.Data_x
8	kostenpflichtige Ressourcen	O.Paid_x
9	Netzwerkkommunikation	O.Ntwk_x
10	Plattformspezifische Interaktionen	O.Plat_x
11	Resilienz	O.Resi_x

Tabelle 1: Auflistung aller Prüfaspekte nach Kategorien[13, p. 16]

4.1 Anwendungszweck

Unter dem Bezeichner O.Purp_x verstehen sich Prüfaspekte, welche sich mit dem Verwenden bzw. Verarbeiten nutzerspezifischer Daten befassen. Die Aspekte sollen sicherstellen, dass Nutzerdaten nur im Sinne des Zwecks der Anwendung verarbeitetet werden. Zusätzlich ist der Nutzer in Form einer Nutzereinwilligung darüber zu informieren, welchem Zweck die Anwendung dient und welche Daten, im Zuge dessen, wie verarbeitet werden. Bestätigte Nutzereinwilligungen sind durch den Betreiber der Anwendung historisch nachvollziehbar und zu jeder Zeit einsehbar zu speichern. Der Nutzer kann den für sich bestehenden historischen Verlauf der Einwilligungserklärungen automatisch und zu jeder Zeit einsehen. Bestehende Einwilligungserklärungen können zu jedem Zeitpunkt widerrufen werden und der Nutzer muss über diese Möglichkeit ausdrücklich informiert werden. In diesem Kontext versteht sich unter Nutzerdaten die Daten des Anwenders und nicht etwa die der jeweiligen Patienten. Da sich die TR des BSI jedoch auf alle Anwendungen im Gesundheitswesen bezieht, muss der Prüfaspekt O.Purp_9 anders ausgelegt werden. Dieser bezieht sich auf das Anzeigen sensibler Daten auf dem Bildschirm. Jedoch geht man hier davon aus, dass der Patient selbst der Anwender ist. Dies ist im Falle dieser Anwendung jedoch nicht so, da es sich um eine Praxisanwendung handelt. Der Zweck des Prüfaspekts wird daher erweitert und bezieht sich im Verlaufe der Arbeit sowohl auf alle auf dem Bildschirm gezeigten Daten des Anwenders, als auch die des Patienten. So ist sichergestellt, dass alle Daten gleichermaßen durch diesen Prüfaspekt abgedeckt sind.

4.2 Architektur

Die Prüfaspekte der Architektur (O.Arch_x) befassen sich mit dem Umgang der Thematik Sicherheit während des Entwicklungsprozesses und der Designphase. Des Weiteren beinhaltet diese Kategorie Vorgaben für das Durchführen gewisser Verfahren, wie Backups und Updates. Insgesamt besteht die Kategorie aus zwölf Aspekten. Jedoch entfallen die Aspekte sechs bis inkl. zwölf. Speziell der Prüfaspekt O.Arch_6 befasst sich mit der Validierung des Anwendungs-Binaries. Eine solche Prüfung ist im Rahmen des Prototyps nicht vorgesehen, daher findet eine Validierung dieses Prüfaspekts im Rahmen dieser Arbeit nicht statt. Sieben und acht entfallen, da die Anwendung weder Drittanbieter-Pakete, noch interpretierten Code wie Javascript nutzt. Die Prüfaspekte neun bis zwölf entfallen, da es sich im Rahmen dieser Arbeit nur um die Betrachtung des Prototyps handelt. Ein Konzept zum Melden von Sicherheitsproblemen oder Nachreichen von Updates ist im Umfang dieses Prototyps nicht vorgesehen und wird daher im Konzept auch nicht beleuchtet.

4.3 Quellcode

Die Kategorie Quellcode (O.Source_x) trifft Vorgaben über die Realisierung des Quellcodes, beispielsweise darüber, wie auf Ausnahmen reagiert werden soll oder welche Methode genutzt werden dürfen. Hierbei entfallen Aspekte im Zusammenhang mit der Entfernung von Entwicklungshilfen (Debug-Code) oder der Entwicklungsumgebung. Dies betrifft die Nummern acht bis zehn. Diese Prüfaspekte gehen davon aus, dass entsprechender Code für die Auslieferung an Kunden gedacht ist. Dies ist jedoch unter Betrachtung eines Prototyps nicht der Fall.

4.4 Drittanbieter-Software

Das Einbinden von Drittanbieter-Paketen kann Arbeit bzw. Zeit bei der Implementierung von Lösungen sparen. Es birgt jedoch auch gewisse Risiken. Diese Risiken werden auch durch das BSI anerkannt. In der TR-03161 unter "Prüfaspekt (4)" des BSI werden entsprechende Maßnahmen/Prüfaspekte (O.TrdP_X) aufgelistet, die nötig sind, wenn Drittanbieter-Pakete verwendet werden.[13, p. 19]

Bei der Entwicklung eines Prototyps bieten sich Open-Source Projekte an. Sie kosten für die Entwicklung nichts und sind je nach Paketverwaltung leicht einzubinden. Soll die Anwendung jedoch eines Tages aus dem Prototyp-Status hinauswachsen und eine kommerzielle Anwendung werden, sind die Lizenzbestimmungen des jeweiligen Pakets sorgfältig zu prüfen.[35] Derartige Bestimmungen werden nicht durch das BSI geprüft und fallen daher, bei einer möglichen Veröffentlichung der Anwendung, in die Verantwortung des Herstellers. Sollten dennoch Drittanbieter-Pakete genutzt werden, ist es ratsam und auch durch das BSI vorgeschrieben eine vollständige Liste der Abhängigkeiten zu führen. Das schließt auch Abhängigkeiten mit ein, die durch ein Paket verursacht werden, bspw. durch das Nutzen eines weiteren Pakets im Paket.[13, p. 19][36]

Während es für das Auflisten der Abhängigkeiten (O.TrdP_1) und regelmäßige Aktualisierungen (O.TrdP_2) etablierte Lösungen gibt, existieren für andere Prüfaspekte im Vergleich keine so praktischen Lösungen. Bspw. muss ein valides Sicherheitskonzept vorliegen, welches den Fall einer Sicherheitslücke in einem Drittanbieter-Paket abdeckt. Des Weiteren ist der Hersteller der Anwendung (nicht der des Pakets) dafür verantwortlich, externe Software auf ihre Sicherheit und Vertraulichkeit zu prüfen. Er muss auch sicherstellen, dass von externen Anwendungen eingehende Daten validiert und sensible Daten nicht mit externen Anwendungen geteilt werden. Drittanbieter-Pakete, die nicht länger gewartet werden, sind gänzlich ausgeschlossen.

Unter Betrachtung der genannten Kriterien ist abzuwägen, ob Drittanbieter-Pakete in diesem Zusammenhang einen Vorteil erzielen oder einen Mehraufwand bedeuten. Eine Auflistung der Abhängigkeiten oder eine vernünftige Update-Politik lassen sich noch mit einfachen Mitteln ermöglichen. Um jedoch Vertrauenswürdigkeit und Sicherheit eines Pakets sicherzustellen, bedarf es einem höheren Zeitaufwand und Know-how als für diesen Prototyp notwendig ist.

Die Sprache, in der dieser Prototyp entwickelt wird, ist C# (C-Sharp). Diese bietet den Vorteil, keine Drittanbieter-Pakete zu benötigen, um kryptografische Verfahren anzuwenden oder API Anfragen zu implementieren.[37] Daher entfallen Prüfaspekte für Drittanbieter-Software (O.Trdp_x) **vollständig**.

4.5 Authentifizierung

Anmerkung: Aufgrund des in dieser Arbeit beschriebenen Authentifizierungsprozesses befinden sich die meisten Prüfaspekte nicht im Rahmen einer Überprüfbarkeit. Dennoch werden im Verlauf solche abgearbeitet. Eine realistische Einschätzung, wie das BSI hier eine Überprüfung durchführen würde, kann in diesem Rahmen nicht durchgeführt werden.

Folgende werden als Prüfaspekte zugelassen, finden jedoch unter Betrachtung des Authentifizierungsprozesses eine andere Anwendung als vorgesehen. So sind bspw. die Aspekte O.Pass_x dafür vorgesehen, die Sicherheit eines Passworts für einen klassischen Login sicherzustellen. In diesem Fall werden diese Aspekte jedoch verwendet, um die Sicherheit des Passwortes sicherzustellen, welches den privaten Schlüssel des jeweiligen Anwenders schützt. In der Analyse wird sich die Authentifizierung in eine lokale und externe Authentifizierung teilen. Wobei eine Anmeldung in die Anwendung lokal und eine Authentifizierung gegenüber dem Hintergrundsystem extern ist. Es werden daher nur Prüfaspekte berücksichtigt, welche sich mit Passwortsicherheit, Sitzungsverwaltung und dem Vorhandensein einer Authentifizierung des Hintergrundsystems befassen. Aspekte zur Thematik Zwei-Faktor-Authentifizierung entfallen.
Auflistung der zugelassenen Prüfaspekte:

1. **Authentifizierung:** O.Auth_6, O.Auth_8 bis O.Auth_11
2. **Passwörter:** O.Pass_1 bis O.Pass_5
3. **Sitzungen:** O.Sess_2 bis O.Sess_6

Die Bezeichner O.Biom_x und O.Tokn_x entfallen vollständig. O.Biom_x bezieht sich auf mobile Anwendungen bzw. Geräte mit einer biometrischen Authentifizierungsmöglichkeit. Die Nutzung von solchen Prozessen ist bei dieser Desktop-Anwendungen jedoch nicht vorgesehen. O.Tokn_x beschreibt den Umgang mit Authentifizierungstoken, diese finden in diesem Konzept jedoch keine Anwendung, auch wenn sie Lösungpotential bieten.

4.6 Datensicherheit

In der Kategorie Datensicherheit befinden sich alle Prüfaspekte mit Bezug zum Datenschutz und Datensicherheit. Hier entfallen alle Bezeichner, die sich explizit auf mobile Endgeräte beziehen (bspw. die Verwendung der Kamera) oder die Sicherheitsmaßnahmen bei Geräte-sperrung prüfen. Das betrifft die Aspekte O.Data_8, bis O.Data_11, O.Data_13, O.Data_14, O.Data_19. Der Prüfaspekt O.Data_16 bezieht sich auf Sicherheitswarnungen bei Übertragung bzw. Speicherung von sensiblen Daten auf Datenträgern, die nicht vor Diebstahl geschützt sind. Dieser Prüfaspekt überschreitet die Anforderungen an den Prototypen, denn die Verantwortung für eine sichere Datenablage liegt laut Anforderung ausschließlich beim Client. Daher entfällt auch dieser Prüfaspekt. Aspekte, die sich mit der Handhabe von Schlüsselmaterial und sensiblen Daten bei bspw. Übertragung oder Deinstallation befassen, werden jedoch berücksichtigt. Ausgenommen davon sind solche, die den Schutz des Schlüsselmaterials durch eine geschützte Umgebung oder Hardware berücksichtigen. Das betrifft O.Data_2 und O.Data_3. Wobei O.Data_2 ausgenommen von Schutzmaßnahmen durch Hardwarekomponenten, berücksichtigt wird. O.Data_3 entfällt jedoch vollständig.

4.7 Kostenpflichtige Ressourcen, Netzwerkkommunikation

Für die Kategorien kostenpflichtigen Ressourcen (O.Paid_x) und Netzwerkkommunikation (O.Ntwk_x) entfallen bis auf O.Ntwk_5 und O.Ntwk_6 alle Prüfaspekte. Es liegen weder bezahlbare Inhalte, noch ein Premium-Modell-Konzept vor. Ein Erwerb der Software ohne Lizenzvereinbarung mit dem Hersteller bzw. Vertreiber der Anwendung ist nicht vorgesehen, daher entfallen die Aspekte in der Kategorie kostenpflichtige Ressourcen. Kapitel *1.3 Zielsetzung und Abgrenzung* grenzt die Prüfaspekte für die Netzwerkkommunikation ein. Im Rahmen dieser Arbeit wird davon ausgegangen, dass die Verbindung vorschriftsmäßig nach TR02102-2[33] eingerichtet bzw. implementiert ist. Sprich: Eine TLS-Verbindung ist eingerichtet und auf dem neusten Stand. Lediglich eine Validierung der Integrität und Authentizität der Nachrichten des Hintergrundsystem muss stattfinden. Das Hintergrundsystem ist nur ein Mittel zur Übertragung und als nicht sicher zu betrachten. Für die Netzwerkkommunikation soll **TLS** in der **Version 1.3** genutzt werden. Daher dürfen auch alle damit verbundenen Funktionen eingesetzt werden.

4.8 Plattformspezifische Interaktionen, Resilienz

Die Prüfaspekte unter der Kategorie plattformspezifischen Interaktionen (O.Plat_x) und Resilienz (O.Resi x) entfallen vollständig. Angesichts dessen, dass es sich hier um die Betrachtung eines Prototyps handelt, führen diese Prüfaspekte zu einer Überschreitung des Rahmens. Spezifisch für den Aspekt O.Resi_6 gilt die Überprüfung der Aspekte aus der Kategorie Netzwerkkommunikation als ausreichend. Daher findet für diesen Punkt speziell keine weitere Argumentation statt.

4.9 Kryptografische Umsetzung

Die Bezeichner O.Cryp_x und O.Rand_x werden in der Kategorie kryptografischen Umsetzung zusammengefasst. Ihre Bedeutung wird im Verlaufe der Analyse gewinnen. Alle in dieser Kategorie angesiedelten Prüfaspekte sind zu berücksichtigen, ausgenommen derer, die sich auf mobile Endgeräte beziehen. Darunter fallen O.Cryp_6 und O.Cryp_7. Zusätzlich ist, aufgrund der Eingrenzung bezüglich der Netzwerkkommunikation, der Aspekt O.Cryp_2 ohne Bezug auf Protokolle im Bereich der Netzwerkkommunikation zu betrachten, lediglich die Vorgabe jederzeit den Standard x509v3 für Zertifikate einzuhalten, wird berücksichtigt. Vorgaben für Zertifikate sind der TR-2103[34] zu entnehmen. Die Prüfaspekte in dieser Kategorie sollen sicherstellen, dass nur kryptografische Verfahren angewendet werden, die auch durch das BSI empfohlen sind. Hierbei ist zusätzlich die TR-02102-1[33] und TR-02102-2[33] zu beachten. Diese technische Richtlinie befasst sich mit kryptografischen Verfahren, deren Eigenschaften und Sicherheit. Sie trifft Aussage darüber, ob und wie ein Verfahren angewendet werden darf. Speziell O.Cryp_4 bezieht sich auf den Umgang mit angewendeten Schlüsseln. Dabei ist zu beachten, dass jeder Schlüssel lediglich eine Aufgabe erfüllt. Primär wird hier der Zweck zum Schutz und zur Authentisierung unterschieden.

Im Folgenden werden die, für den Prototyp wichtigsten, Verfahren grob erläutert und gegeneinander aufgewogen.

4.9.1 Schlüssellänge symmetrischer Verschlüsselung

Für eine symmetrische Verschlüsselung wird ausschließlich der "Advanced Encryption Standard" (AES) empfohlen. Aus vergangener Version der Richtlinie wurden Serpent und Twofisch gestrichen, da diese in Vergangenheit keiner ausreichender Prüfung unterzogen wurden. Für AES wird eine Schlüssellänge von mindestens 128 Bit empfohlen.[33, p. 27]

4.9.2 Initialization Vector (IV)

Der bei einigen symmetrischen Verfahren verwendete "Initialization vector" (IV) wird genutzt, um zu verhindern, dass eine verschlüsselte Nachricht einer anderen auf gleichen Klartext basierenden chiffrierten Nachricht identisch ist, auch wenn ein identischer Schlüssel verwendet wird. Kurz: Der IV verhindert, dass eine Verschlüsselung deterministisch ist.[21] Den IV zu verschlüsseln ist im allgemeinen nicht nötig, solange der entsprechende Schlüssel geheim bleibt.[38, 39] Es gibt jedoch in anderen Umgebungen der direkten Netzwerkkommunikation Schwachstellen, die auf einem offenen IV basieren. Voraussetzung für diese Schwächen ist unter anderem die Vorhersage des nächsten IV Werts.[40, 41] Daher ist für die Generierung des Schlüssels und des IV der Abschnitt *4.9.10 CSPRNG* und die damit zusammenhängenden Prüfaspekte O.Rand_1 bis O.Rand_4 besonders zu beachten. Ein 2020 veröffentlichtes Paper legt nahe den IV genauso zu schützen, wie den zusammenhängenden Schlüssel, um die Integrität der Nachricht samt IV sicherzustellen.[42]

Die hier beschriebenen Probleme basieren jedoch nur auf ähnlichen Arbeitsweisen, wie bspw. einer Kommunikation mit ständig wechselnden IV bei gleichbleibendem Schlüssel. Eine Sicherstellung der Integrität ist notwendig, die Verschlüsselung des IV jedoch nicht.

4.9.3 Schlüssellänge asymmetrischer Verfahren

Das BSI empfiehlt in der TR-02102-1 lediglich drei asymmetrischer Verfahren für Verschlüsselung bzw. einen Schlüsselaustausch: **ECIES**, **DLIES** und **RSA**. Die Minimallänge eines Schlüssels unterscheidet sich je nach Verfahren.[33, p. 32] Grundsätzlich benötigen *"klassische"* Verfahren wie RSA im Vergleich zu Verfahren basierend auf elliptischen Kurven wesentlich längere Schlüssel, um ein gleiches Sicherheitsniveau zu erreichen. So ist die Schlüssellänge von 1024 Bits des RSA-Verfahrens äquivalent zu einer Länge von 139 Bits für Verfahren mit elliptischen Kurven. Zusätzlich steigt die Schlüssellänge bei elliptische Kurven für ein steigendes Sicherheitsniveau wesentlich langsamer.[43, p. 101-102] Daher gelten für die Verfahren DLIES und RSA eine minimale Schlüssellänge von 2000 Bits, für das ECIES-Verfahren jedoch nur eine minimale Schlüssellänge von 250 Bits.[33, p. 32]

Anmerkung: Für RSA gilt die angegebene Schlüssellänge nur bis Ende des Jahres 2023. Mit Ende des Jahres 2023 wird eine minimale Schlüssellänge von 3000 Bits empfohlen. Für DLIES ist eine Schlüssellänge von 3000 Bits jedoch bereits mit Ende des Jahres 2022 vorgesehen. Die Schlüssellänge des ECIES-Verfahrens bleibt jedoch vorerst bestehen.[33, p. 32]

4.9.4 Betriebsarten symmetrischer Verschlüsselung (Blockchiffre)

Das BSI empfiehlt für Blockchiffren vier Betriebsarten: **CBC**, **CTR**, **CCM** und **GCM**. Die letzteren beiden (CCM, GCM) sind Verfahren für "Authenticated Encryption" (AE), oder auch "Authenticated Encryption with Associated Data" (AEAD). Dies sind Algorithmen, die sowohl Verschlüsselung, als auch Datenintegrität und Datenauthentizität sicherstellen.[33, 12, p. 29, p. 308, 315] Die AE/AEAD Betriebsarten CCM und GCM werden in Abschnitt *4.9.5 Betriebsarten symmetrischer Verschlüsselung (AE/AEAD)* näher behandelt.

Unter den empfohlenen Betriebsarten taucht die klassische Blockchiffre **"Electronic Code Book" (ECB)** nicht mehr auf. ECB arbeitet im Gegensatz zu CBC und CTR ohne den Zusatz eines IV. Das hat zur Folge, dass bei gleichem Klartext und gleichem Schlüssel der Ciphertext grundsätzlich identisch ist. Das bedeutet: Besitzt ein potenzieller Angreifer den Schlüssel für einen Ciphertext, so ist es ihm möglich, alle identischen Ciphertexte zu entschlüsseln, ohne dafür erneut einen Schlüssel zu erarbeiten. Aufgrund der deterministischen Eigenschaft dieses Vorgehens, ist es durch Häufigkeitsanalyse möglich, Teile des Klartextes zu rekonstruieren.[33, p. 28] Zusätzlich werden beim ECB-Verfahren alle zu verschlüsselnden Blöcke unabhängig voneinander verschlüsselt. Das bedeutet: Ein Angreifer kann gezielt einzelne Blöcke manipulieren, ohne dabei den Rest der verschlüsselten Nachricht zu beeinflussen.

Die Betriebsart **"Cipher Block Chaining (CBC)"** arbeitet im Gegensatz zum ECB-Verfahren mit einem IV welcher ein deterministisches Vorgehen verhindern soll. Sprich: Das Resultat der Verschlüsselung ist nicht nur vom Schlüssel abhängig, sondern zusätzlich vom Wert des IV. Wie der Name bereits suggeriert, arbeitet dieses Verfahren mit zusammenhängenden Blöcken. Der zuvor verschlüsselte Block beeinflusst den darauffolgenden Block. Ein Austausch oder eine Manipulation einzelner Blöcke ist daher so ohne weiteres nicht mehr möglich. Der IV ist bei diesem Vorgehen lediglich für den ersten zu verschlüsselnden Block wichtig, denn dieser hat keinen vorangehenden Block, der ihn beeinflusst.[12, 21, p. 308-311, p. 65-66] Eine Verschlüsselung des ersten Blocks ohne IV würde die selben Schwächen zulassen, wie das ECB-Verfahren. Durch die Abhängigkeit der verschlüsselten Blöcke ist es dem Empfänger beim Entschlüsseln möglich, vertauschte oder entfernte Blöcke zu erkennen. Zusätzlich ist es möglich Fehler, und damit auch eventuell gezielte Manipulation in einzelnen Blöcken zu erkennen.[22, p. 95-97]

Anmerkung: Unter Anwendung der Betriebsart CBC ist zu beachten, dass ein entsprechendes Padding gemäß der BSI-TR-02102-1[33, p. 30] anzuwenden ist. Die Eigenschaften einzelner Paddings werden hier jedoch nicht beleuchtet und auch im Verlauf nicht betrachtet.

Der **"Counter Mode" (CTR)** arbeitet, wie das CBC-Verfahren, mit einem IV. Dieser wird jedoch nicht, wie bei CBC, ausschließlich für den ersten Block genutzt. Pro verschlüsseltem Block wird ein Zähler (Counter) hochgezählt, welcher in Kombination mit dem Wert des IV genutzt wird, um einen "neuen" Vektor zu erhalten, welcher für den nächste Block genutzt wird. Es entsteht ein Strom-Ciphertext, welcher unabhängig von den vorangegangenen verschlüsselten Blöcken chiffriert wird. Das bedeutet, bei entfernten oder vertauschten Blöcken entsteht ein Fehler in der Zählung des Counters, wodurch die Entschlüsselung fehlschlägt. Bei einem Bit-Fehler betrifft dies im Verlauf der Entschlüsselung nur den jeweiligen Block, nicht jedoch die nachkommenden. Derartige Fehler werden also nicht direkt durch den Empfänger erkannt.[21, 22, p. 69-70, p. 97-89]

4.9.5 Betriebsarten symmetrischer Verschlüsselung (AE/AEAD)

Betriebsarten der Kategorie "Authenticated Encryption" (AE) oder auch "Authenticated Encryption with Associated Data" (AEAD) bieten die Möglichkeit mit symmetrischer Verschlüsselung sowohl Vertraulichkeit, als auch Authentizität und Integrität von Daten sicherzustellen. Dies geschieht durch kombinierte Anwendung einer Verschlüsselung und einem "Message Authentication Code" (MAC).[12, p. 315][44] Der wesentliche Unterschied zwischen den genannten Kategorien AE und AEAD ist die Verwendung von "Associated Data" (AD). AE-Verfahren nutzen ausschließlich eine Kombination aus der zu verschlüsselnden Nachricht und dem symmetrischen Schlüssel. Während AEAD-Verfahren zusätzlich nicht zu verschlüsselnde Zusatzinformationen anwenden, um Authentizität und Integrität aller zu sendenden Daten, sowohl verschlüsselte als auch nicht verschlüsselte (AD), sicherzustellen.[45] Das BSI empfiehlt zwei Verfahren: "Counter with Cipher Block Chaining Mode" (CCM) und "Galois Counter Mode" (GCM). Beide Verfahren sind in der Lage mit AD zu arbeiten und zählen daher in die Kategorie AEAD.[46, 47, 48, 49]

"Counter with Cipher Block Chaining Mode" (CCM) basiert auf der Kombination aus CBC-MAC und einer CTR-Chiffre. Auch hier ist, angesichts der kombinierten Verfahren, ein IV notwendig. Zunächst wird aus dem Klartext, welcher später chiffriert werden soll, und den zusätzlichen Daten (AD) eine CBC-MAC erzeugt. Unter Anwendung des CTR-Verfahrens wird daraufhin die zu verschlüsselnde Nachricht inklusive der MAC chiffriert. Dieses Vorgehen nennt sich "mac-then-encrypt". Das bedeutet die Authentizität und Integrität kann erst nach dem Entschlüsseln der Nachricht mit Hilfe der MAC überprüft werden.[46, 47, 50]

Der **"Galois-Counter-Modus" (GCM)** arbeitet Blockweise und benötigt wie CCM einen IV. Auch GCM arbeitet mit einer CTR-Chiffre, welche jedoch für dieses Verfahren leicht variiert. Die MAC zur Sicherstellung der Authentizität und Integrität wird hier mit einem eigenen Verfahren unter verwendung des sogenannten Galoiskörpers erzeugt (GMAC).[12, 21, p. 316, p. 255] Anders als bei CCM arbeitet GCM mit "encrypt-then-mac", was es dem Empfänger erlaubt, zuerst Authentizität und Integrität sicherzustellen, bevor eine vollständige Entschlüsselung stattfindet.[50, p. 6]

Anmerkung: Das BSI empfiehlt für die Erzeugung einer MAC drei Vorgehensweisen: **CMAC**, **HMAC** und **GMAC**. Alle Verfahren sollten mit eines minimalen Schlüssellänge von 128 Bit erzeugt werden und in einer Taglänge von 96 Bit resultieren.[33, p. 50] Bei der Erzeugung einer HMAC müssen die Vorgaben in Bezug auf Hash-Algorithmen beachtet werden. Siehe Abschnitt *4.9.8 Hash-Algorithmen*.

4.9.6 Asymmetrische Ver- und Entschlüsselung

Das BSI empfiehlt drei asymmetrische Verfahren: **ECIES**, **DLIES** und **RSA**. Wie im Abschnitt *4.9.3 Schlüssellänge asymmetrischer Verfahren* erläutert, sind Schlüssel auf Basis von elliptischen Kurven wesentlich kürzer. ECIES und DLIES sind im Vergleich zu RSA ein speziell hybrides Verschlüsselungsverfahren. Diese Verfahren sind spezifisch für den Einsatz eines Schlüsselaustauschs gedacht. Laut BSI gibt es jedoch keine relevanten Unterschiede im Bereich der Sicherheit innerhalb dieser Verfahren, vorausgesetzt die Schlüssellänge ist auf Basis der Sicherheit auf ein Äquivalent angepasst (ECIES = 250 Bits, DLIES/RSA = 3000 Bits).[33, p. 31]

Es gibt jedoch, im Gegensatz zu ECIES und DLIES, Voraussetzungen bei der Anwendung von RSA. Unter Anwendung von RSA ist das klassische **Schema PKCS#1.5** nicht mehr in der Liste der empfohlenen Verfahren. Das Problem - PKCS#1.5 ist deterministisch. Sprich: Bei gleicher Eingabe und gleichem Schlüssel folgt auch eine gleiche Ausgabe als Ciphertext. Dadurch können aus einem Ciphertext Informationen über den Klartext gewonnen werden.[12, p. 340-341][51] Stattdessen soll auf das **Schema EME-OAEP** gesetzt werden. Ähnlich wie bei AES mit CBC werden hier zusätzliche Informationen verwendet, um einen NICHT deterministischen Ablauf zu erzeugen.[52]

4.9.7 Digitale Signaturen

Das BSI empfiehlt eine Reihe an Signaturverfahren, darunter **RSA**, **DSA** und DSA-Varianten, auf Basis von elliptischen Kurven (**ECDSA**). Wie bereits im Abschnitt *4.9.3 Schlüssellänge asymmetrischer Verfahren* beschrieben, gelten auch hier die entsprechenden Schlüssellängen.

Ein Signaturverfahren auf Basis von elliptischen Kurven hat den Vorteil, bei seiner Erstellung weniger Ressourcen (Ressourcen $\neq$ Geschwindigkeit) zu benötigen.[12, p. 347] Jedoch bedarf eine klassische Signatur (RSA) bei der Auswertung weniger Ressourcen.[53] Signaturen auf Basis von elliptischen Kurven (speziell ECDSA) werden in vielen Bereichen eingesetzt, in denen Sicherheit die höchste Rolle spielt, bspw. der digitale neue Personalausweis (nPA).[12, p. 347-348] Jedoch konnte bisher nicht festgestellt werden, dass das RSA-Verfahren kritische Schwächen aufweist.[53] Im Geschwindigkeitsvergleich hat DSA, bei Erstellung von Signaturen, den Vorteil. Bei der Verifikation dieser liegt jedoch RSA deutlich vorn. Der ECDSA liegt, ausgenommen bei der Schlüsselkonstruktion, zeitlich bei beiden Vorgängen hinten.[54]

In Vergangenheit wies der DSA und ECDSA, in Bezug auf Implementierung in OpenSSL und Debian, Schwächen bei schlecht erzeugten Zufallszahlen und Seitenkanalangriffen auf.[55, 56, 57, 58] Es ist daher bei Anwendung dieser Verfahren besonders darauf zu achten, wie die Implementierung durchgeführt wird. Aufgrund der beim DSA benötigten Zufallszahl, die bei jeder Signatur erzeugt wird, ist es dringend notwendig, entsprechend sichere Generatoren zu nutzen.[33, p. 51-52]

Anmerkung: Bei der Erstellung von Signaturen ist es notwendig, einen Hashwert zu erzeugen, dieser muss unter Berücksichtigung der geltenden Vorgaben der TR gewählt werden. Siehe *4.9.8 Hash-Algorithmen*. Zusätzlich sollte bei der Verwendung von RSA darauf geachtet werden, ein entsprechendes, vom BSI empfohlenes Formatierungsverfahren zu wählen (**EMSA-PSS**).

4.9.8 Hash-Algorithmen

Das BSI empfiehlt ausschließlich Hash-Algorithmen aus der Familie "Secure Hash Algorithms" (SHA). Ausgenommen davon ist SHA-1, da dieser in Vergangenheit keine ausreichende Kollisionsresistenz nachweisen konnte. Es werden grundsätzlich nur Algorithmen empfohlen, die ein minimales Sicherheitsniveau von 120 Bits aufweisen können. Wobei hier wie üblich unter SHA SHA2 zu verstehen ist.

Daraus ergeben sich folgende Empfehlungen:
SHA-256, **SHA-512/256**, **SHA-384** und **SHA-512**
SHA3-256, **SHA3-384**, **SHA3-512**
[33, p.47]

4.9.9 Passwortbasierte Schlüsselableitung

Für die Passwortbasierte Schlüsselableitung empfiehlt das BSI das Generieren einer MAC auf Basis des Passworts. Als Empfehlung wird hier speziell CMAC und HMAC genannt. Laut den Vorgaben muss diese MAC jedoch durch eine Hardwarekomponente mit Zeitverzögerung generiert werden. Dies wird der Einfachheit halber in dieser Arbeit nicht vorausgesetzt.[33, p. 79-80] Mit welchen Verfahren die HMAC oder CMAC erzeugt wird, ist nicht speziell beschrieben, lediglich dass eine Hardwarekomponente verwendet werden soll. Ist das Verwenden einer kryptografischen Sicherheitskomponente nicht möglich, so wird Argon2id empfohlen. Dies ist das einzige, in diesem Zusammenhang genannte Verfahren.

Ein mögliches Verfahren auf Basis von HMACs ist PBKDF2. PBKDF2 bietet die Möglichkeit, eine HMAC zu erzeugen, welche mit einer selbst festgelegten Anzahl von Iterationen generiert wird. Die Anzahl der Iterationen ist ausschlaggebend für die Entschleunigung eines Brute-Force-Angriffs.[12, p. 446] Das BSI gibt jedoch keine Anzahl an Iterationen vor. Das NIST hingegen empfiehlt minimal 1.000 Iterationen. Für starke Systeme oder Hochrisikowerte wird ein Wert von bis zu 10.000.000 empfohlen.[59, p. 6] Die Anzahl der Iterationen ist vor allem dann wichtig, wenn Passwörter abgeleitet werden, die bspw. Dateien schützen, welche leicht aus der Anwendungsumgebung entfernt werden können. Unter Anwendungsumgebung versteht sich z.B. ein Endgerät in einem Büro. Ist es einem theoretischen Angreifer möglich, die geschützte Datei leicht in seine eigene Umgebung zu transferieren, ist die Anzahl der Iterationen der einzige Faktor zur Entschleunigung des Angriffs.

PBKDF2 weist besondere Schwächen bei "custom hardware attack"/"application-specific integrated circuits" (ASIC) oder parallelen Angriffen mit bspw. GPUs auf.[60] Diese Schwäche soll die gesondert erwähnte Alternative des BSI, Argon2id (Argon2) nicht mehr aufweisen. Argon2 ist der Gewinner der *'Password Hashing Competetion' 2015*[61]. Wie PBKDF2 bietet auch Argon2 die Möglichkeit, Iterationen einzustellen, um Brute-Force-Angriffe zu entschleunigen. Zusätzlich dazu kann auch der Speicherverbrauch angegeben werden, um parallele Angriff zu erschweren. Diese Maßnahmen sollen die von PBKDF2 bekannten Schwächen ausgleichen.[62, 63, 64]

Anmerkung: .NET setzt für die Erzeugung der HMAC mit PBKDF2 SHA-1 ein.[65] Das BSI weist daraufhin, dass SHA-1 in Vergangenheit durch Kollisionsprobleme aufgefallen ist und deshalb diese Art der Hashwert-Bildung nicht mehr genutzt werden sollte. Jedoch schließt es SHA-1 nicht für eine HMAC-Konstruktion aus. Es wird jedoch dazu geraten auf diese Familie der Hashwert-Konstruktion zu verzichten.[33, p. 47]

4.9.10 Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

Wie bereits in voran gegangenen Abschnitten erläutert ist es von besonderer Bedeutung starke Zufallszahlen zu erzeugen und zu nutzen. Das BSI gibt mehrere Lösungen zu diesem Problem an. In diesem Abschnitt wird sich primär auf die Lösung nicht-physikalische nicht-deterministische Zufallszahlengeneratoren (NPTRNG [NTG.1]) fokussiert, denn unter der Definition für NPTRNG wird auf ein weiteres Dokument (Siehe [66]) des BSI verwiesen, welches detailliert Testcharakteristika und weitere Definitionen offenlegt. Dieses Dokument nennt unter der Definition von NPTRNG Microsoft Windows als explizites Beispiel.[66, p. 20] Zusätzlich werden die Testcharakteristika mit Hilfe des Standards "Federal Information Processing Standards 140-2" (FIPS 140-2) definiert.[66, p. 44] Grundsätzlich sind die wichtigsten kryptografischen Methoden

unter der Verwendung eines aktuellen Windows Betriebssystems (min. Windows 7) FIPS 140-2 zertifiziert.[67, 68] Der Standard FIPS 140-2 wird durch das NIST definiert und angewendet. Der Standard definiert Grundlagen und Prüfungen für die Implementierung von kryptografischen Algorithmen und deren Module. Das schließt die Erzeugung von CSPRNG mit ein.[69]

.NET (und damit C#) kann standardmäßig Zufallszahlen erzeugen. Dies geschieht über die Klasse "RandomNumberGenerator" (RNG). Diese Klasse ist ein kryptografischer Zufallszahlengenerator.[70] Wird bspw. die Klasse AES genutzt, um Schlüssel und IV Werte zu erzeugen, wird auf den RNG zurückgegriffen. .NET bietet bei der Anwendung von bspw. AES drei Variationen an: [*CryptoServiceProvider] (CSP), [*Cng] (Cryptography Next Generation [CNG]) und [*Managed]. Auf die Nutzung von [*Managed] sollte, speziell in diesem Rahmen verzichtet werden, da für diese Klasse keine Zertifizierung für den "Federal Information Processing Standards" (FIPS) vorliegt. Der CSP und CNG sind Wrapper-Klassen und greifen auf systemspezifische Algorithmen zurück. Dabei werden FIPS-zertifizierte Implementierungen bevorzugt.[71] Das bedeutet .NET greift unter Anwendung dieser Klassen auf die Windows eigenen Bibliotheken zurück, um Zufallszahlen zu erzeugen. In neueren Systemen wird dafür "CNG.SYS" eingesetzt. Speziell wird auf die Funktion "BCryptGenRandom" zurückgegriffen. Der RNG erzeugt gleich aus einer Vielzahl von Entropien eine Zufallszahl.[72]

In diesem Rahmen und nach der Definition des BSI ist anzunehmen, dass der durch .NET standardmäßig angesprochene Zufallszahlenenerator ausreichend ist, um die im Rahmen dieses Prototyps erzeugten Schlüssel zu verwenden. Gerade im Hinblick auf die Tatsache, dass die System-Methoden von Windows auf eine kryptografische Stärke mit Hilfe des FIPS 140-2 geprüft wurden.[67, 68] Es ist jedoch darauf zu achten, dass der, in dieser Arbeit beschriebene, Prototyp ausschließlich im Rahmen einer aktuellen Windows-Version arbeiten darf, um diese durch FIPS 140-2 geprüften Module/Methoden nutzen zu können.

Anmerkung: Die Klasse [*CryptoServiceProvider] sollte auf lange Sicht nicht verwendet werden. Sie baut auf der "Cryptographic Application Programming Interfaces" (CAPI) auf.[73] Das CAPI wird auf lange Sicht durch CNG ersetzt.[74, 75]

4.10 Übersicht

Nr.	Kategorie	Bezeichner
1	Anwendungszweck	O.Purp_1 - O.Purp_9 *(ALLE)*
2	Architektur	O.Arch_1 - O.Arch_5
3	Quellcodes	O.Source_1 - O.Source_7
4	Drittanbieter-Software (third party)	***ENTFALLEN***
5	Authentifizierung: Passwörter, Biometrischedaten, Sitzungen und Tokens	O.Auth_6 und O.Auth_8 - O.Auth_11 O.Pass_1 - O.Pass_5 *(ALLE)*, O.Sess_2 - O.Sess_6
6	Datensicherheit	O.Data_1, O.Data_2, O.Data_4 - O.Data_7, O.Data_12, O.Data_15, O.Data_17 und O.Data_18
7	Kostenpflichtige Ressourcen	***ENTFALLEN***
8	Netzwerkkommunikation	O.Ntwk_5 und O.Ntwk_6
9	Plattformspezifische Interaktionen	***ENTFALLEN***
10	Resilienz	***ENTFALLEN***
11	Kryptografische Umsetzung	O.Cryp_1 - O.Cryp_5 O.Rand_1 - O.Rand_4 *(ALLE)*

Tabelle 2: Auflistung aller zugelassenen Prüfaspekte [13, p.16]

5 Prototyp

5.1 Anforderung und Design

Die Anwendung soll eine sichere Datenübertragung innerhalb einer Auswertegemeinschaft zur Verfügung stellen. Das bedeutet, ein Auswerter (bspw. ein Kardiologe) soll einen Auftrag mit Rohdaten zur Auswertung durch einen gesicherten Übertragungsweg erhalten können. Ein Auswerter kann diese Rohdaten auswerten und das Ergebnis sicher an den entsprechenden Auftraggeber zurückgeben. Aufträge werden von verpartnerten Hausarztpraxen zugewiesen, daher werden diese Hausarztpraxen als Zuweiser bezeichnet. Die zu übertragenen Dateien bestehen grundsätzlich aus einer GDT-Datei und einer PDF-Datei oder Rohdaten. Wobei Rohdaten inkl. GDT-Datei üblicherweise durch einen Zuweiser versendet werden und durch einen Auswerter ausgewertet werden. Eine Auswertung auf Basis dieser Rohdaten besteht dabei aus der GDT-Datei inkl. einer PDF-Datei, welche bspw. den Befund beinhaltet. Aus Gründen der Speicherplatzersparnis und der Hoch- und Herunterlade-Geschwindigkeit werden die Dateien vor dem Hochladen komprimiert. Es ergibt sich eine ungefähre, durchschnittliche Dateigröße von ∼3 MB. Der Ablauf von einer Zuweisung von Rohdaten bis hin zum Herunterladen der Auswertung dieser Zuweisung wird als Prozess oder Vorgang bezeichnet.

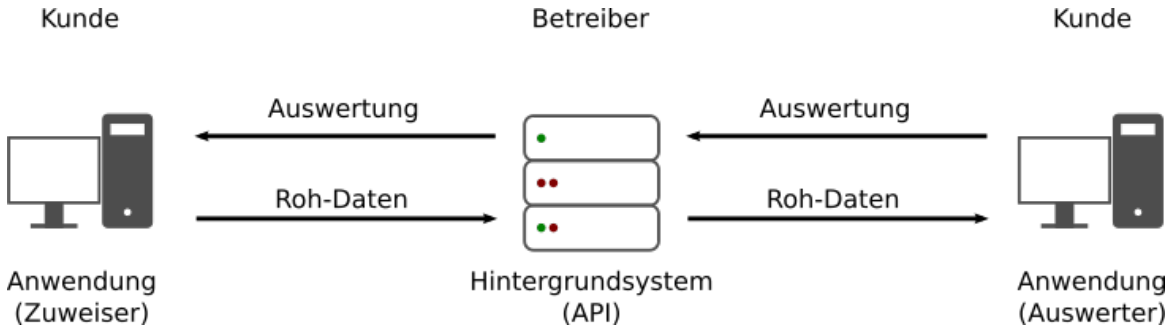

Abbildung 1: Grobe Konzeptbetrachtung

Der Server (Hintergrundsystem) dient hierbei als Mittelsmann und ist in der Lage, Aufträge zur Übertragung entgegenzunehmen und entsprechende Dateien temporär zu lagern. Das Hintergrundsystem soll durch die Anwendung nur als Mittel zum Zweck gesehen werden und daher nicht als vertrauenswürdig betrachtet werden. Daher darf das Hintergrundsystem die auf ihm abgelegten Daten zu keinem Zeitpunkt im Klartext enthalten. Es ist wichtig anzumerken, dass das Hintergrundsystem zu keinem Zeitpunkt in der Lage sein darf, Daten, die einem Patienten zuzuordnen sind, zu entschlüsseln oder anderweitig zu betrachten bzw. zu verarbeiten. Lediglich die einzelnen Praxen dürfen in der Lage sein, die zu ihnen passenden Patientendaten zu entschlüsseln und zu betrachten. Die Anwendung ist dafür verantwortlich, zu transportierende Daten ausschließlich verschlüsselt zu übertragen, so dass in der Datenablage und Datenbank des Hintergrundsystems sensible Daten ausschließlich chiffrierte hinterlegt sind. Ausgenommen davon sind Daten, die zwingend für die Zuordnung der Daten/Dateien seitens des Hintergrundsystems notwendig sind. Gleichzeitig muss die Anwendung Daten, die sie herunterlädt, auf Integrität und Nachweisbarkeit prüfen können. Für die Verschlüsselung von Daten/Dateien soll die Anwendung ein symmetrisches Verfahren verwenden. Für den Schutz der symmetrischen Schlüssel, sowie zur

Garantie der Integrität und Nachweisbarkeit von Daten, soll entsprechend ein asymmetrisches Verfahren verwendet werden, wobei hier zu beachten ist, dass für Signaturen und Verschlüsselung zwei unterschiedliche asymmetrische Schlüssel zu verwenden sind. Möchte ein Client ein Paket an einen Kommunikationspartner übermitteln, muss er dafür Sorge tragen, dass sowohl die Daten, als auch die Dateien signiert und verschlüsselt sind und die dazu gehörigen symmetrischen Schlüssel ebenfalls geschützt sind. Dazu muss der Sender die jeweiligen öffentlichen Schlüssel seines Kommunikationspartners durch das Hintergrundsystem erhalten. Diese muss er unabhängig des Hintergrundsystems prüfen können. Nur wenn die öffentlichen Schlüssel des Empfängers verifiziert sind, dürfen mit diesen Signaturen verifiziert oder die symmetrischen Schlüssel mit diesem chiffriert werden. Daten, die durch einen Nutzer heruntergeladen wurden und die Anwendung verlassen, liegen grundsätzlich in der Verantwortung des Nutzers. Die Anwendung ist nicht für die lokale Sicherheit von Dateien verantwortlich. Hat ein Zuweiser Dateien hochgeladen, muss es ihm möglich sein, seine entsprechenden Dateien wieder herunterzuladen und zu entschlüsseln. Dasselbige gilt auch für den Auswerter und seine Auswertungen. Um es einem Anwender oder Zuweiser möglich zu machen mehrere Datenpakete für sich zu identifizieren, ohne dabei die eigentlichen Dateien herunterzuladen, müssen die Dateien getrennt von Daten zur "Vorschau" zur Verfügung stehen.

Für eine sichere Verbindung zwischen Client und Server wird TLS-1.3 eingesetzt. Entsprechend steht dem Betreiber eine vertrauenswürdige "Certification Authority" (CA) zur Verfügung. Es dürfen ausschließlich Zertifikate verwendet werden, die den Standard x509v3 erfüllen. Nach dem Verbindungsaufbau muss sich der Client gegenüber dem Hintergrundsystem ident- und authentifizieren. Dazu soll ein Authentifizierungsverfahren genutzt werden, welches keine Passwortanforderung besitzt (mutual authetifikation).

Das Hintergrundsystem besteht primär aus einer Datenbank und einem File-Storage. Um diese Dienste für eine Übertragung zu nutzen, kann das System via API erreicht werden. Die Datenbank beinhaltet dabei alle nötigen Relationen, um Kommunikationspartner miteinander in Verbindung zu bringen. Der File-Storage erfüllt lediglich die einfache Aufgabe der Datenablage. Das Hintergrundsystem ist ebenfalls dafür verantwortlich, dass jeder Client nur die Prozesse zu sehen bekommt, die sich auch an ihn richten oder von ihm ausgehen.

5.1.1 Schlüsselpaar

Die Anforderungen machen es erforderlich, dass jeder Nutzer, entsprechend dem asymmetrischen Verfahren, zwei Schlüsselpaare besitzt. Soll die Anwendung auf mehreren Geräten einer Praxis verfügbar sein, so ist es notwendig bzw. die einfachste Lösung, alle Geräte mit den gleichen Schlüsselpaaren zu versorgen. Daher soll ein Container geschaffen werden, welcher alle nötigen Einstellungen und Daten inkl. der Schlüsselpaare enthält. Dieser Container soll als XML-Datei realisiert werden. Die Schlüsselpaare dürfen ausschließlich verschlüsselt in dieser Datei gelagert werden. Ein entsprechender Schlüssel für diese Verschlüsselung, soll von einem nutzerspezifischen Passwort abgeleitet werden.

5.1.2 Einwilligungserklärung

Möchte ein Auswerter oder Zuweiser die Anwendung nutzen, benötigen beide die Anwendung und einen entsprechenden Lizenzvertrag mit dem Betreiber. Ein Lizenzvertrag geht immer mit einer Einwilligungserklärung einher. Diese klärt den Anwender darüber auf, welche Daten wie

verwendet werden. Da Nutzerdaten sich in diesem Zusammenhang auf die Daten der jeweiligen Praxis beziehen, wird diese im Lizenzvertrag darauf hingewiesen, dass alle Patienten, deren Daten mit Hilfe dieser Anwendung transportiert werden, darüber zu informieren sind, welche ihrer Daten wie und wozu genutzt werden. Ein Lizenzvertrag, sowie die damit zusammenhängende Einwilligungserklärung, wird digital, noch vor der eigentlichen Installation durch den Betreiber, eingefordert. Die jeweilige Praxis ist in der Lage, diese digitale Einwilligung jeder Zeit zurückzuziehen. Eine Nutzung der Anwendung ist danach jedoch nicht mehr möglich.

Per Definition sind Praxisnamen nicht direkt personenbezogen, da die Daten lediglich einen Rückschluss auf ein Unternehmen bzw. eine Praxis liefern und nicht direkt auf eine lebende Person schließen. Dieser Umstand ändert sich jedoch, wenn Praxisnamen den Namen des Arztes oder der Ärzte beinhalten. Deshalb erfordert die Speicherung und Nutzung der genannten Daten immer einer Zustimmung nach DSGVO aller Praxen. Dabei ist auch klar aufzuführen, wie diese Daten verwendet werden.

5.1.3 Einstellungen

Bei der Übertragung eines Prozesses müssen, für die Identifikation seitens des Nutzers, Patientendaten aus der GDT-Datei gelesen werden. Das betrifft die folgenden Eigenschaften:

Feldkennung	**Eigenschaft**	**Anmerkung**
3000	GDT-Patientennummer-Zuweiser	NICHT eindeutig
3102	Vorname	NIE NULL
3101	Nachname	NIE NULL
3103	Geburtsdatum	NIE NULL DD.MM.YYYY
3100	Namenspräfix	Von, Zu
3104	Titel	Dr., Prof.

Tabelle 3: Verwendete Patientendaten in den Prozessdaten

Diese Daten sind notwendig, um dem Nutzer beim Betrachten eines Prozesses die Möglichkeit zu bieten, den Patienten zu identifizieren, ohne die damit verbundenen Dateien herunterzuladen. Bei der Erstellung eines Prozesses wird jedoch bei der Auswahl der Dateien in zwei Fälle unterschieden. Die Anwendung läuft entweder im GDT-ONLY-Modus oder GDT-EXCLUDET-Modus. Ist der GDT-ONLY-Modus aktiv, zwingt die Software den Nutzer dazu Dateien auszuwählen, die auch eine valide GDT-Datei enthalten. Für den GDT-EXCLUDET-Modus gilt diese Vorgabe nicht. Die unter den Dateien vorhandene GDT-Datei wird genutzt, um Patientendaten zu erheben. Läuft die Anwendung im GDT-EXCLUDET-Modus und es ist keine GDT-Datei vorhanden, ist der Nutzer gezwungen, die Patientendaten manuell einzutragen.

Der GDT-ONLY-Modus ist der Standard-Modus, da ein Erlauben einer manuellen Eingabe von Patientendaten Sicherheitsrisiken bergen oder menschliches Versagen begünstigen kann. Obwohl Nutzereingaben gegen gängige Fehler/Risiken, wie bspw. zu langen Eingaben oder Eingaben mit Sonderzeichen, geschützt sind (Escaped), wird hier davon ausgegangen, dass das

Laden der GDT-Datei weniger Gefahren birgt. Entspricht die GDT-Datei nicht dem vorgegebenen Standard, wird der Prozess mit einem Fehlerfall abgebrochen. Auch hier werden alle Zeilen der Datei geschützt (Escaped) gelesen.

5.1.4 Kryptografische Algorithmen

Die folgenden Argumentationen basieren auf dem Wissensstand, welcher in Kapitel *4.9 Kryptografische Umsetzung* beschrieben wird. Alle symmetrischen und asymmetrischen Schlüssel werden, wenn nicht anders beschrieben, durch die Standardfunktionen von .NET (Cryptography Next Generation) erzeugt. Nach dem beschriebenen Wissensstand ist dies für die Durchführung dieser Anwendung ausreichend.

Key-Container: Um die in Abschnitt *5.1.1 Schlüsselpaar* genannten Schlüsselpaare zu schützen, muss durch eine Passwortableitung zunächst ein Schlüssel gewonnen werden. Dazu wird **PBKDF2 mit 10.000 Iterationen** verwendet. Das in Kapitel *4.9 Kryptografische Umsetzung* genannte Verfahren Argon2 ist in .NET nicht standardmäßig umgesetzt. Daher kommt, aufgrund der Einschränkung von Drittanbieter-Paketen, nur das Verfahren PBKDF2 in Frage. Es wird unter Anwendung des Verfahrens ein Schlüssel der Länge 32 Byte (256 Bit) abgeleitet. Für die Verschlüsselung der Schlüsselpaare selbst sollte, nach Vorgaben des BSI, nur AES eingesetzt werden. Um unbemerkte Änderungen im Container, bspw. an den Einstellungen, zu verhindern, sollte ein AEAD-Verfahren genutzt werden. Hier kann **GCM mit einer Schlüssellänge von 256 Bit** genutzt werden, um noch vor dem eigentlichen Entschlüsseln die Integrität der AD zu prüfen. Als AD sollten unter anderem der, im Verlaufe der Implementierung beschriebene, Nutzeridentifikator und die IV/Salt-Werte der Container-Verschlüsselung und der Passwortableitung verwendet werden.

Asymmetrisches Verfahren: Für den Prototyp sind in diesem Kontext zwei asymmetrische Schlüsselpaare vorgesehen. Zunächst wird ein Verfahren für Ver- und Entschlüsselung gewählt. EC-Verfahren sind allein durch ihre Schlüssellänge und ihren Einsatz in Systemen mit hohem Sicherheitsniveau im Vorteil. Jedoch besitzt C# keine Standardimplementierung für die empfohlenen Schlüsseltransportverfahren ECIES und DLIES, weshalb lediglich das klassische **RSA-Verfahren mit dem Schema EME-OAEP** und einer **Schlüssellänge von 3000 Bit** zur Verfügung steht. Für Signaturverfahren existieren alle Implementierungen, um ein beliebiges, nach den Richtlinien des BSI empfohlenes, Verfahren auszuwählen. Namentlich besitzt .NET Implementierungen für RSA, DSA[76] und ECDSA[77]. Aufgrund der geringen Schlüssellänge wird hier auf **ECDSA mit 250 Bit Schlüssellänge** gesetzt. Ein weiterer Abgleich zwischen DSA und anderen Verfahren entfällt, da die Unterstützung für diese Implementierung bereits zurückgezogen wurde.[76] Mit den gewählten Schlüssellängen ist eine Sicherheit über das Jahr 2023 hinaus gewährleistet.

(A) Anmerkung: Im Verlauf der folgenden Kapitel wird der Begriff elliptische Kurven mit EC abgekürzt. Ein EC-Public-Key bezieht sich also auf die Verwendung eines öffentlichen Schlüssels auf Basis von elliptischen Kurven.

(B) Anmerkung: Auf die Problematik mit "Encrypt-then-Sign" und "Sign-then-Encrypt" wird im Rahmen dieser Arbeit nicht weiter eingegangen. Es sei an dieser Stelle auf [78] verwiesen. Zumindest für Dateien ist das Verfahren "Sign-Encrypt-Sing" anzuwenden.

Symmetrisches Verfahren: Um die Sicherheit der Dateien und Daten zu gewährleisten, soll ein symmetrisches Verschlüsselungsverfahren (AES) eingesetzt werden. Dazu ist zunächst abzuwiegen, ob ein "klassisches" Verfahren (CBC, CTR) oder ein AEAD-Verfahren (CCM, GCM) genutzt werden soll. Dateien und Daten sollen auf ihre Integrität und Nachweisbarkeit überprüfbar sein. Eine reine Integritäts- und Authentizitätsprüfung, wie sie AEAD-Verfahren gewährleisten, sind genau deshalb nicht ausreichend, da das Hintergrundsystem - als nicht vertrauenswürdiges System - von unbefugten Dritten einen Prozess erhalten könnte, der angesichts der vollständigen Schöpfung durch den Angreifer als legitim angesehen werden kann. Es bedarf also Nachweisbarkeit. Zusätzlich muss die Anwendung in der Lage sein Prozessdaten vor der Vorschau für den Nutzer zu validieren. Daher ist nicht nur ein unabhängiges Herunterladen von den Dateien notwendig, sondern auch eine separate Signatur. Die Prüfung auf Integrität und Nachweisbarkeit von Daten/Dateien soll also unabhängig voneinander stattfinden können. Angesichts dessen ist eine Verwendung eines AEAD-Verfahrens hier nicht notwendig. Daher rücken die Betriebsarten CBC und CTR in den Fokus. Da C# keine Standardimplementierung für CTR besitzt, fällt die Entscheidung auf CBC mit einer Schlüssellänge von 256 Bit.

Zertifikate: Um öffentliche Schlüssel von Kommunikationspartnern verifizieren zu können, müssen diese als Zertifikat (Nutzerzertifikat) durch das Hintergrundsystem übergeben werden. Jeder Client besitzt für die unabhängige Verifizierung dieser Zertifikate zusätzlich ein lokales Zertifikat (Serverzertifikat) welches eine Verifizierung möglich macht. Alle Zertifikate erfüllen grundsätzlich den Standard x509v3. Das Serverzertifikat ist, für einen unabhängige Prüfung, durch eine CA ausgestellt. Die Nutzerzertifikate hingegen sind durch den privaten Schlüssel des Betreibers signiert. Das Serverzertifikat ist das Gegenstück zu diesem Schlüssel. Nutzerzertifikate können also durch das Serverzertifikat validiert werden.

Anmerkung: Der private Schlüssel ist, angesichts der Vertrauensfrage, nicht auf dem eigentlichen Server bzw. Hintergrundsystems gelagert. Er muss unabhängig des Servers sicher aufbewahrt werden.

Verfahren	**Einsatzgebiet**
PBKDF2 (10.000 Iterationen)	Schlüsselableitung Key-Container
AES-256 (GCM)	Verschlüsselung Key-Container
EC-Schlüsselpaar (250 Bit)	ECDSA Signaturen
RSA-Schlüsselpaar (3000 Bit)	EME-OAEP Verschlüs. sym. Schlüssel
SHA-256 (SHA2)	Hashwert-Generierung (Signaturen)
AES-256 (CBC)	Verschlüsselung von Prozessdaten u. Dateien
x509v3 (Serverzertifikat)	Validierung von Nutzerzertifikaten
x509v3 (Nutzerzertifikate)	Zur Verfügung stellen der öff. Schlüssel

Tabelle 4: Angewendete Verfahren in Übersicht und Funktion

5.2 Konzept

5.2.1 Einrichtung

Die Installation der Anwendung wird grundsätzlich durch einen Techniker durchgeführt. Vor der Installation legt der Techniker im Hintergrundsystem der Anwendung den neuen Nutzer an. Dazu legt er zunächst in Kommunikation mit dem neuen Nutzer einen Identifikator fest. Der Identifikator wird in Verbindung mit dem Praxisnamen, der Kundennummer und der Rolle in der Datenbank abgelegt. Bei der darauffolgenden Ausführung bzw. dem ersten Start der Anwendung wird der Identifikator und die Kundennummer eingegeben. Die Anwendung prüft über eine Abfrage, ob in der Datenbank ein entsprechender Eintrag vorliegt. Ist diesem so - generiert die Anwendung ein RSA- und EC-Schlüsselpaar, bestehend aus dem privaten und öffentlichen Schlüssel. Die öffentlichen Schlüssel werden dem Techniker zur Verfügung gestellt, damit dieser diese zertifizieren kann. Die Schlüsselpaare und der Identifikator werden daraufhin in den verschlüsselten Container eingefügt und der Nutzer muss unabhängig vom Techniker ein eigenes Passwort für die Schlüsselableitung festlegen. Findet eine Deinstallation der Anwendung statt, wird dieser Container vernichtet.

Die Stärke des Passworts zum Schutze des Containers ist nutzerabhängig. Wählt der Nutzer ein schwaches Passwort, ist die Sicherheit des Containers gefährdet. Daher muss dem Nutzer suggeriert werden, dass sein Passwort eine entsprechende Stärke aufweisen muss. Dazu wird während der Passworteingabe die entsprechende Stärke dargestellt und ein zu schwaches Passwort nicht zugelassen. Zusätzlich wird der Nutzer über die Folgen, bei einem möglichen Passwortverlust, informiert. Aufgrund des Authentifizierungsprozesses gegenüber des Hintergrundsystems mit Hilfe des Containers hat ein solcher Verlust Folgen, da sich der Container ohne Passwort nicht entschlüsseln lässt. Ohne diesen kann das Hintergrundsystem keine Ident- oder Authentifizierung sicherstellen und entsprechende geschützte symmetrische Schlüssel können nicht entschlüsselt werden. Ist das Passwort eingegeben, kann ein Nutzer ein neues Passwort festlegen. Dabei gelten verständlicherweise dieselben Bestimmungen, wie bei dem ersten Passwort. Der Container wird infolge der Änderung mit dem neuen Passwort bzw. mit dessen Ableitung chiffriert. Eine Kopie des Containers in Klartext ist nicht zulässig. Es bleibt bei Verlust des Passworts keine andere Möglichkeit, als den Container mit neuem Passwort vollständig neu zu generieren. Prozesse, die mit dem ursprünglichen Container gestartet bzw. erstellt wurden, sind aufgrund der, mit dem verlorenen Key-Pair, geschützten, symmetrischen Schlüssel nicht mehr lesbar. Auch die im Hintergrundsystem hinterlegten Zertifikate müssen, aufgrund der neu generierten Schlüsselpaare, erneuert werden.

Das bedeutet, im Hintergrundsystem befindet sich für jeden Nutzer zwei Zertifikate, welche durch den privaten Schlüssel des Servers signiert wurden. Das Serverzertifikat ist jedoch extern durch eine CA signiert. Ein Nutzer besteht dem zufolge aus folgenden Eigenschaften:

Eigenschaft	**Anmerkung**
Kundennummer	
Identifikator	Kurz: Ident
Praxisname	
Zertifikate	RSA- und EC-Public-Key
Rolle	Auswerter oder Zuweiser

Tabelle 5: Nutzereigenschaften im Hintergrundsystem

Der Ident, die Kundennummer und die Zertifikate sind als sensibel eingestuft. Der Ident wird als Identifikation der Anwendung gegenüber dem Hintergrundsystem verwendet. Anderen Nutzern soll es daher nicht möglich sein, einen Ident eines anderen Nutzers zu erfahren. Nutzer, die einander über eine Relation im System kennen, können sich anhand des Praxisnamens erkennen. Das bedeutet: Fordert bspw. ein Zuweiser die ihm zugeordneten Auswerter an, erhält dieser nur die Praxisnamen und die Zertifikate zur Verifizierung der öffentlichen Schlüssel, jedoch keine weiteren Identifikationsmittel.

Die für Nutzer erhobenen Daten halten sich bewusst im begrenzten Rahmen. Der Praxisnamen wird zur Identifikation der Nutzer untereinander und der Ident zur getrennten Identifikation gegenüber des Hintergrundsystem verwendet. Die Kundennummer kann innerhalb des Vertriebs bspw. genutzt werden, um unterzeichnete Lizenzbestimmungen oder AGBs zuzuordnen. Zusätzlich wird bei der Installation das Serverzertifikat des Betreibers mitgeliefert. Dieses dient als Authentifizierungsmittel für bspw. Zertifikate anderer Nutzer. Es ist das Gegenstück zu dem privaten Schlüssel, welcher die Nutzerzertifikate signiert.

5.2.2 Authentifizierungsprozess

Die Schlüsselpaare (Key-Pairs) erfüllen gleich mehrere Aufgaben. Zum einen dient das RSA-Paar der sicheren Übertragung von symmetrischen Schlüsseln, welche beim Versenden eines Prozesses entstehen. Das EC-Schlüsselpaar kann, für Signaturen und eine passwortlose Authentifizierung eines Nutzers gegenüber des Hintergrundsystems genutzt werden. Hierzu wird zwischen zwei unterschiedlichen Authentifizierungsmitteln unterschieden: Lokale Authentifizierungsmittel, welche genutzt werden, um den Nutzer an **seinem Endgerät** zu Authentifizieren und externe Authentifizierungsmittel, welche genutzt werden, um den Nutzer gegenüber des Hintergrundsystems zu ident- und authentifizieren. Das lokale Authentifizierungsmittel ist das Passwort, welches ausschließlich dem Schutz der externen Mittel bzw. der Schlüsselpaare dient. Die externen Mittel befinden sich entsprechend in dem verschlüsselten Container. Nur durch eine Eingabe des Passworts lässt sich der Container freigeben.

Möchte der Nutzer die Anwendung nach erfolgreicher Installation nutzen, so wird bei jedem Start zunächst das Serverzertifikat validiert und der Anwender wird dazu aufgefordert sein Passwort einzugeben. Mit diesem wird der entsprechende Container auf Integrität geprüft und entschlüsselt. Die nun offen liegenden Authentifizierungsmittel (Ident. und Schlüsselpaare) können für die automatische und Passwortfreie, externe Authentisierung genutzt werden.

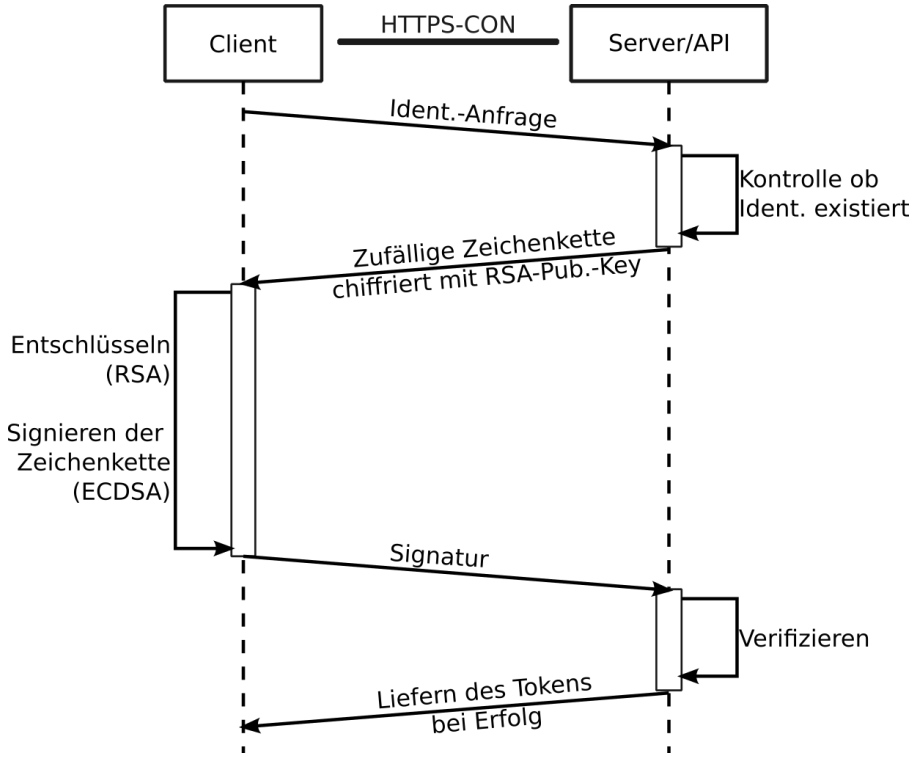

Abbildung 2: Authentifizierungsprozess (Client-Server)

Der eigens entwickelte Authentifizierungsprozess basiert auf dem Grundgedanken einer SSH-Public-Key-Authentication (mutual authentication). Damit ist der hier beschriebene Ablauf sehr ähnlich dem einer SSH-Public-Key-Authentication[11, p. 308-312]. Der Authentifizierungsprozess setzt voraus, dass vor der Authentifizierung bereits eine sichere TLS-Verbindung aufgebaut wurde und lediglich eine Client-zu-Server-Authentifizierung aussteht. Hierzu schickt der Client seinen Ident. an das Hintergrundsystem. Dieses prüft, ob der Ident. so in der Datenbank zu finden ist. Ist diese Prüfung erfolgreich, antwortet das Hintergrundsystem dem Client mit einer zufälligen Zeichenkette, verschlüsselt mit dem, auf dem Server hinterlegten, öffentlichen Schlüssel, des zu authentifizierenden Nutzers (Challenge). Die Zeichenkette wird vom Client mit dem RSA-Schlüssel entschlüsselt. Infolgedessen wird die Zeichenkette mit SHA-256 zu einem Hashwert verarbeitet. Dieser wird mit dem, im Container hinterlegten, EC-Schlüsselpaar des Clients signiert (ECDSA) und darauffolgend dem Hintergrundsystem übermittelt. Das Hintergrundsystem generiert auch, auf Basis der Zeichenkette, einen entsprechenden Hashwert. Um die Validierung abzuschließen, wird dieser Hashwert und das für diesen Client entsprechend hinterlegte Zertifikat genutzt, um die vom Client übermittelte Signatur zu prüfen. Ist die Validierung erfolgreich, bekommt der Client ein Token übermittelt, welches für zukünftige Operationen genutzt werden kann.

Der Client speichert dieses Token jedoch nur während seiner Laufzeit ab. Erfolgt ein Neustart oder ein Beenden der Anwendung verfällt dieses Token seitens des Clients und des Hintergrundsystems. Befindet sich der Client länger als 10 Minuten im "Idel-Modus" so wird der Container automatisch verschlüsselt und das Token verfällt. Eine neue Eingabe des Passworts wird darauf gefordert, gefolgt durch eine erneute Anmeldung gegenüber dem Hintergrundsystem. Im Hintergrundsystem behalten Tokens ebenfalls maximal 10 Minuten ohne Tätigkeiten des Clients ihre Gültigkeit, oder bis zu einer Abmeldung seitens des Clients. Läuft ein Token ab,

fordert das Hintergrundsystem eine erneute Authentifizierung.

Die Sicherheit dieses Verfahrens basiert nahezu allein auf der kryptografischen Eigenschaft der genutzten asymmetrischen Verfahren, in diesem Fall ECDSA und RSA. Ist es einem zu authentifizierenden Client möglich, eine valide Signatur anhand der verschlüsselten zufälligen Zeichenkette zu erzeugen, muss er folglich im besitzt beider privaten Schlüssel sein, welche zu den hinterlegten Zertifikaten passen. Da die Kommunikation über eine HTTPS-Verbindung stattfindet, ist sichergestellt, dass kein unbefugter Dritter in der Lage ist, das Token oder die Zeichenkette abzufangen. Es ist unter Anwendung von ECDSA Signaturen besonders auf die Stärke der Zufallszahlen zu achten, auch dann, wenn eine TLS-Verbindung besteht. (Siehe *4.9.7 Digitale Signaturen*)

5.2.3 Prozessablauf und Übertragung

Entsprechend der Anforderung liegt der Kernpunkt der Anwendung im Übertragen von Rohdaten oder ausgewerteten Daten. Eine vollständige Übertragung wird als Vorgang oder auch Prozess bezeichnet. Das bedeutet: Zuweiser können Vorgänge erstellen, Auswerter können diese auswerten und Zuweiser können ausgewertete Vorgänge herunterladen. Ein Prozess besteht dabei aus zwei voneinander abhängigen Bestandteilen: Den Prozessdaten und den Prozessdateien. Prozessdaten sind Daten, die den Prozess als solchen ausmachen und identifizieren. Bspw. werden Prozessdaten für eine Vorschau genutzt, damit der Nutzer einen Prozess erkennen kann, ohne dabei Dateien herunterzuladen. Kurzum handelt es sich um all die Daten, die in der Datenbank abgelegt werden. Die Prozessdateien sind Dateien, die den eigentlichen, zu transportierenden Anteil ausmachen. Diese Dateien umfassen sowohl die vom Zuweiser zur Verfügung gestellten Rohdaten inkl. GDT-Datei zur Auswertung, als auch die Auswertung des Auswerters als PDF-Datei inkl. GDT-Datei. Sie werden nur heruntergeladen, wenn der Nutzer dies wünscht. Prozessdaten bestehen ausfolgenden Eigenschaften:

Eigenschaft	**Anmerkung**
Vorgangsnummer	Eineindeutige ID
Patientendaten	Siehe Tabelle 3
Ident-Sender	
Ident-Empfänger	
Status	Übermittelt, Heruntergeladen, Ausgewertet, Erledigt u. Korrektur
Erstellungsdatum	
Datum der letzten Änderung	
AES-Schlüsselmaterial	Siehe Tabelle 7
Signatur-Dateien	Nach der Verschlüsselung
Signatur-Daten	Nach der Verschlüsselung

Tabelle 6: Prozessdaten

Bezeichnung	**Anmerkung**
AES-KEY-Zuweiser	Verschl. mit Public-Key des Zuweisers
AES-KEY-Auswerter	Verschl. mit Public-Key des Auswerters
AES-IV-Wert	Unverschlüsselt

Tabelle 7: Prozessdatenschlüsselmaterial

Die Vorgangsnummer ist eine eineindeutige Zahl zur eindeutigen Zuordnung des Prozesses. Der Ident-Sender bzw. Ident-Empfänger ist der Identifikator für die jeweilige Praxis. Fordert eine Praxis, die zu ihr gehörenden Vorgänge, an wird dieser immer durch den jeweiligen Praxisnamen ersetzt. Der Status gibt die Phase in der sich der Prozess befindet an. Eigenschaften wie *Erstellungsdatum* und *Datum der letzten Änderung* sprechen für sich. Eine Erläuterung der Eigenschaften *AES-Schlüssel* und *Signatur-Dateien* folgt im Verlauf dieses Kapitels. Die Eigenschaften der Patientendaten sind der *Tabelle 7 Prozessdatenschlüsselmaterial* zu entnehmen.

Anmerkung: Die *Signatur-Daten* werden mit dem gleichen Verfahren erzeugt wie *Signatur-Dateien*, jedoch mit dem Unterschied, dass die zugrundeliegenden, zu signierenden Daten andere sind. In diesem Fall handelt es sich um die Prozessdaten, darunter fallen: *Patientendaten*, *Ident-Sender*, *Ident-Empfänger*, *Erstellungsdatum*, *AES-KEY-Zuweiser*, *AES-KEY-Auswerter* und *AES-IV-Wert*.

Als schützenswert, und damit verschlüsselt, werden alle personenbezogenen Daten eingestuft. Allem voran gilt das für die zu transportierenden Dateien, denn diese enthalten den größten Anteil sensibler Daten. Darunter zum Beispiel alle beim Prozess genutzten Patientendaten, Diagnosen und Rohdaten bzw. kardiologische Aufzeichnungen eines Patienten. Die Prozessdaten werden jedoch nur zum Teil verschlüsselt. Grundsätzlich werden alle Patientendaten inkl. der *GDT-Patientennummer-Zuweiser*, verschlüsselt. Die *GDT-Patientennummer-Zuweiser* hat einen reinen lokalen Nutzen seitens des Zuweisers. Eine Patientennummer einer GDT-Datei ist lediglich die Zuordnung innerhalb einer Praxis. Das bedeutet: Eine Patientennummer ist nur innerhalb einer Praxis zuordenbar und hat damit außerhalb dieser Praxis keine Bedeutung. Zusätzlich zu personenbezogenen Daten müssen auch die Schlüssel geschützt werden, die genutzt werden, um die personenbezogenen Daten zu verschlüsseln. Mehr dazu im Abschnitt *5.2.5 Kryptografisches Protokoll*.

Ein Vorgang durchläuft im Verlaufe seiner Abarbeitung - sprich: Erstellung bis zum Herunterladen der Auswertung - vier bis fünf Phasen. Bei Erstellung des Vorgangs wird standardmäßig der Status *Übermittelt* gesetzt. Für den Prozess bedeutet das: Der Vorgang besitzt alle nötigen Daten in der Datenbank und die Dateien wurden ordnungsgemäß hochgeladen. Auf den Status *Übermittelt* folgt immer der Status *Heruntergeladen*. Das bedeutet: Der Auswerter hat aus seiner Liste den Prozess ausgewählt und die Dateien heruntergeladen. Darauf folgt zwingend der Status *Ausgewertet*: Der Auswerter hat eine für diesen Prozess getätigte Auswertung fertiggestellt und zur Verfügung gestellt. Der Status *Ausgewertet* kann nur durch das Herunterladen der Auswertung seitens des Zuweisers auf *Erledigt* geändert werden. Ist ein Vorgang als *Erledigt* markiert, so kann der Auswerter optional eine *Korrektur* hochladen, wodurch der Status auf *Korrektur* springt. Auf eine *Korrektur* folgt wieder ein Abschluss des Vorgangs (*Erledigt*). Befindet sich der Vorgang im Status *Ausgewertet* können Korrekturen hochgeladen werden,

ohne den Status zu beeinflussen.

Grundsätzlich gilt, dass der Status sich nur nach Erfolg der jeweiligen Operationen verändert. Das bedeutet zum Beispiel, dass der Vorgang nur als erledigt gilt, wenn der Zuweiser auch die Daten vollständig und korrekt erhalten hat. Schlägt bspw. die Validierung der Dateien fehl, so verändert sich der Status nicht. Wird ein Prozess gelöscht, so werden alle mit ihm zusammenhängenden Daten und Dateien entfernt bzw. im Falle von Dateien vernichtet. Das betrifft auch symmetrische Schlüssel.

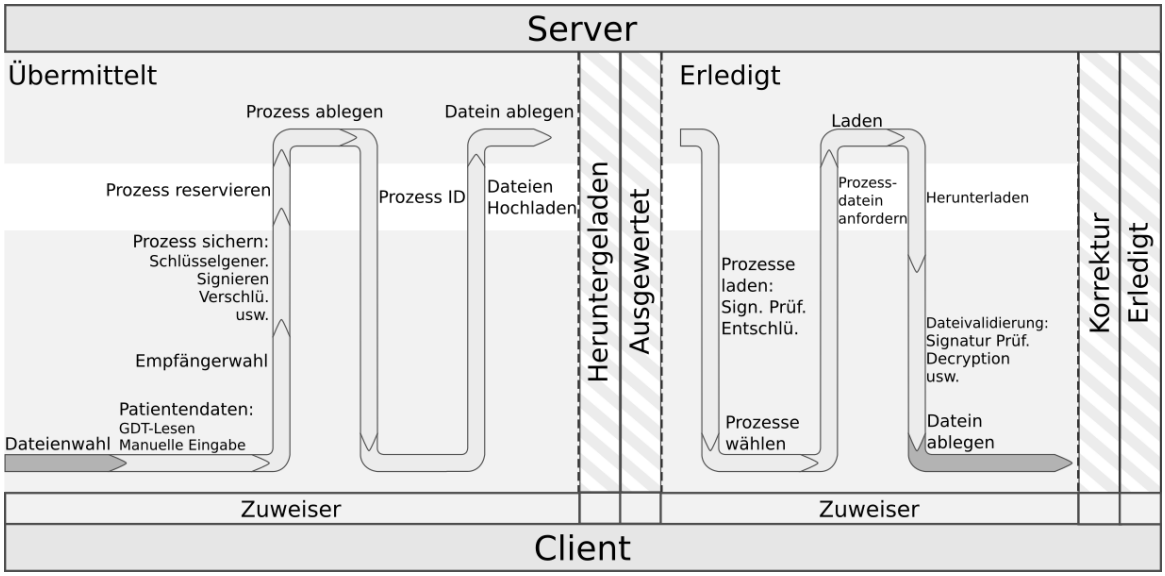

Abbildung 3: Prozesszyklus aus Sicht des Zuweiser

In der Sicht des Zuweisers beginnt der Prozess nur durch seinen Wunsch. Er wählt zunächst die gewünschten, zu transportierenden Dateien aus. Je nach Betriebsmodus wird daraufhin die GDT-Datei ausgelesen oder die nötigen Patientendaten werden manuell eingetragen. Ist die GDT-Datei fehlerhaft bricht die Prozesserstellung ohne weitere Operation ab. Ist dem nicht so, kann der Zuweiser einen Empfänger auswählen. Darauf folgt die Sicherung des Prozesses. Der Ablauf zur Prozesssicherung und zum Laden der Empfängerdaten wird im Abschnitt *5.2.5 Kryptografisches Protokoll* detaillierter beschrieben.

Vor dem eigentlichen Ablegen bzw. Hochladen der Dateien benötigt der Client jedoch eine aktuell verfügbare Vorgangsnummer. Dafür reserviert der Client zunächst einen Prozess durch das Übermitteln der Prozessdaten zur API. Diese antwortet mit einer gültigen Vorgangsnummer, welche im Anschluss genutzt wird, um die Prozessdateien zu übermitteln.

Ist ein Prozess als ausgewertet markiert, kann der Zuweiser die Auswertung herunterladen. Hierfür wählt er in der Prozessliste den jeweiligen Prozess aus und lädt die dazugehörigen Dateien herunter. Die Anwendung validiert nach dem Herunterladen die Dateien und entschlüsselt sie.

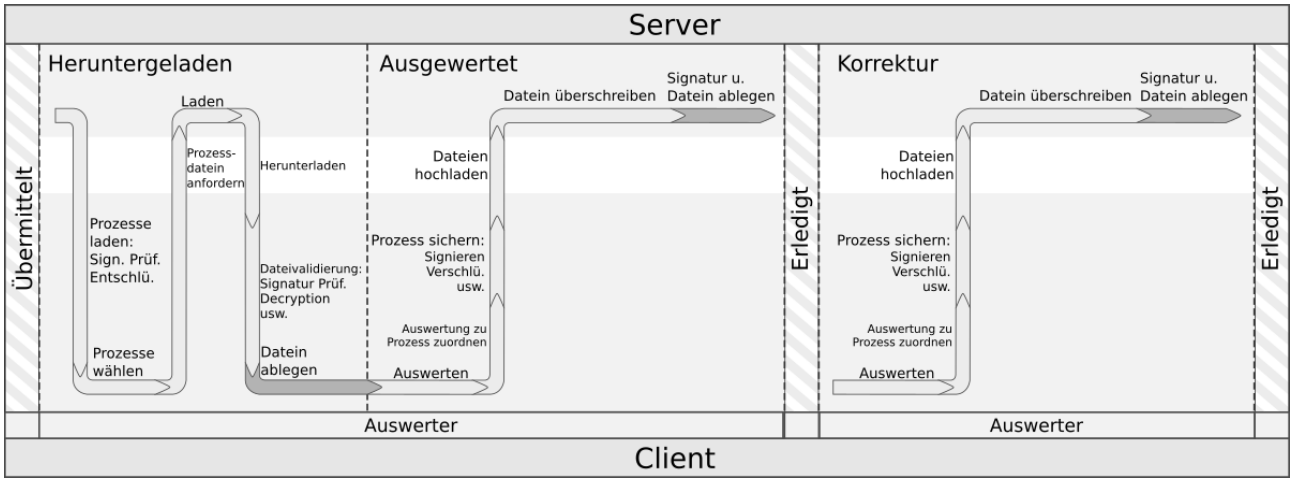

Abbildung 4: Prozesszyklus aus Sicht des Auswerters

Aus Sicht des Auswerters unterscheiden sich nur ein Vorgehen von den Operationen eines Zuweisers. Das Prozedere beim Herunterladen von Prozessen ist mit denen des Zuweisers identisch. Das Hochladen einer Auswertung unterscheidet sich jedoch in dem Punkt, dass hier vorher kein Empfänger geladen und validiert werden muss. Der Auswerter muss lediglich seine Auswertung mit dem dazugehörigen Prozess in Verbindung bringen. Die Anwendung erstellt daraufhin für die neuen Dateien eine neue Signatur. Beim Hochladen der Dateien wird auch die neue Signatur mitgeliefert. Das Hintergrundsystem überschreibt die alten Dateien mit den neuen und ersetzt die alte Signatur durch die neue. Das Hochladen einer Korrektur verläuft identisch.

Während dieser Abläufe kann jede der beiden Parteien zu jeder Zeit einen Vorgang vernichten und damit die Dateien sowie die Prozessdaten vernichten. Im Zustand *Übermittelt* kann sowohl der Zuweiser als auch der Auswerter die Dateien immer wieder anfordern bzw. herunterladen. Das gilt auch für einen erledigten Zustand nur, dass hier der Auswerter seine Auswertung erhält und nicht die ursprünglichen Rohdaten zum Beginn des Vorgangs, denn diese wurden entsprechend überschrieben.

5.2.4 Datenlebenszyklus

Beim Transport der Dateien ist stets die Sicherheit der Dateien zu betrachten. Zu keinem Zeitpunkt darf es zu dem Umstand kommen, dass unbefugte Dritte Dateien erhalten, im Klartext lesen oder sogar manipulieren können. Es ist daher vonnöten, zu jeder Phase den Überblick zu behalten. Dazu dient ein Datenlebenszyklus. Dieser stellt graphisch jede Phase und Operation der genutzten Dateien dar.

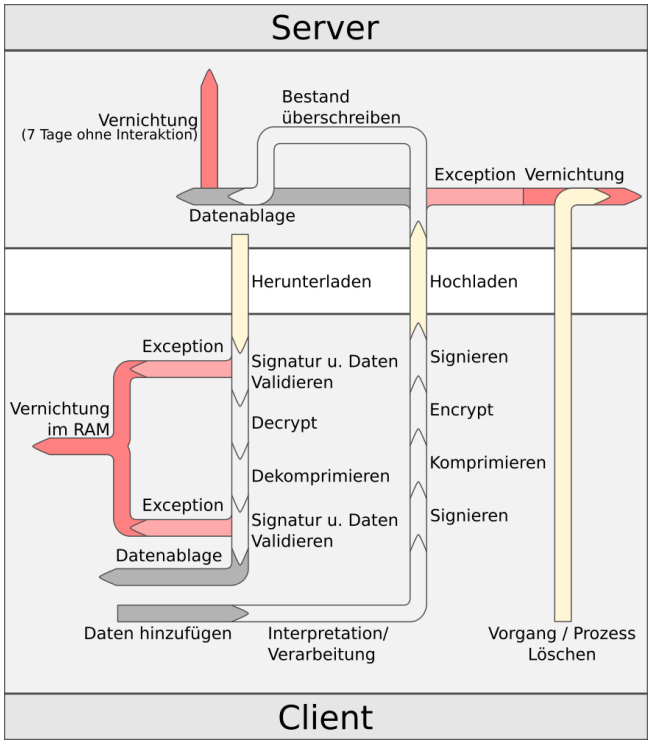

Abbildung 5: Datenlebenszyklus

Um Dateien in den Zyklus zu integrieren, müssen diese, wie bereits in den vorangegangenen Abschnitten beschrieben, vom Nutzer ausgewählt werden. In der Verarbeitung wird die GDT-Datei ausgewertet. Ist diese nicht vorhanden, entfällt dieser Schritt. Ist der Upload ausgelöst, werden zunächst Hashwerte für die einzelnen Dateien erstellt. Die entstandenen Hashwerte werden im Anschluss mit Hilfe des nutzerspezifischen privaten Schlüssels zu einer Signatur. Die Signaturen sowie die Dateien werden daraufhin zu einer Datei komprimiert (ZIP). Diese wird dann mit dem, für diesen Prozess generierten AES-256-Schlüssel, chiffriert und ebenfalls signiert. Nun kann das Hochladen stattfinden. Das Hintergrundsystem entscheidet nach dem Erhalten der Dateien, ob es bestehende Dateien vor dem Ablegen überschreiben muss oder ob für diesen Prozess noch keine Dateien vorliegen. Tritt im Hintergrundsystem bei bspw. der Validierung* der Dateien ein Fehler auf oder die Dateien liegen mehr als sieben Tage unberührt auf dem System, werden diese restlos vernichtet.

**Das Hintergrundsystem ist durch den Besitzt der nutzerspezifischen Zertifikate der öffentlichen Schlüssel in der Lage, Signaturen unabhängig des Clients zu validieren. Ein Fehlschlag dieser Validierung löst einen Fehlerfall aus.*

Beim Herunterladen von Dateien wird nach dem vollständigem Empfangen der Dateien zunächst die Signatur validiert, um die Nachweisbarkeit und Integrität sicherzustellen. Erst nachdem dies fehlerfrei geschehen ist, werden die Dateien mit dem AES-Key entschlüsselt und dekomprimiert. Die entschlüsselten, einzelnen Dateien werden dann nochmals, durch validieren der Signatur, auf Nachweisbarkeit und Integrität geprüft. Kommt es hierbei zu keiner Ausnahme, legt die Anwendung die Dateien im Zielverzeichnis ab. Der einzige Sinn und Zweck des Herunterladens ist es, die Dateien unverschlüsselt im Zielverzeichnis abzulegen und dabei in keiner Form zu verarbeiten oder anzuzeigen. Mit der Ablage der Dateien im Zielverzeichnis verlassen diese den

Verantwortungsbereich der Anwendung. In einem Ausnahmefall, zum Beispiel durch eine nicht valide Signatur, bricht der Vorgang sofort ab und vernichtet die Dateien durch Überschreiben im Arbeitsspeicher.

5.2.5 Kryptografisches Protokoll

Das kryptografische Protokoll beschreibt detailliert die Verwendung kryptografischer Algorithmen im Zusammenhang mit den bereits erläuterten Vorgehensweisen. Es soll nochmals aufgezeigt werden, welche Verfahren genutzt werden und wofür. Die Veranschaulichung einer Prozesserstellung inkl. Empfängervalidierung soll das Zusammenspiel der Verfahren verdeutlichen. Es ist anzumerken, dass alle verwendeten Verfahren inkl. ihrer Verwendung der *Tabelle 4 Angewendete Verfahren in Übersicht und Funktion* zu entnehmen sind.

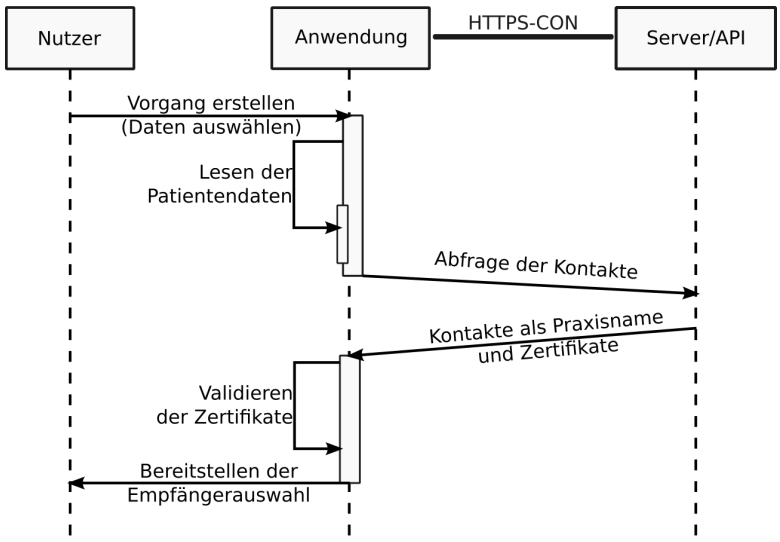

Abbildung 6: Kontaktdaten Empfangen und Validieren

Die Anwendung muss für die Empfängerauswahl von dem Hintergrundsystem alle Kontakte bzw. alle möglichen Empfänger für diesen Nutzer erhalten. Dafür bedarf es einer Kontaktanfrage. Diese wird meist bei Erstellung eines Prozesses automatisch oder durch den Nutzer selbst manuell ausgelöst, ohne einen Prozess zu erstellen. Liefert das Hintergrundsystem die jeweiligen Kontakte so bestehen diese aus dem Praxisnamen und den beiden Zertifikaten (RSA-Pub.-Key und EC-Pub.-Key). Die Zertifikate beinhalten die öffentlichen Schlüssel des jeweiligen Kontakts. Mit dem bei der Installation mitgelieferten Zertifikat des Servers werden alle erhaltenen öffentlichen Schlüssel verifiziert. Kommt es dabei zu einem Fehler, wird der Nutzer entsprechend darauf hingewiesen und es ist ihm unter keinen Umständen möglich, dem jeweiligen Kontakt einen Vorgang zuzuweisen. Ist ein Zertifikat nicht erfolgreich verifiziert, besteht das Risiko, dass unbefugte Dritte den öffentlichen Schlüssel des Kontakts manipuliert haben. Das Hintergrundsystem muss bei entsprechenden Fehlern in Kenntnis gesetzt werden, damit die Administration Maßnahmen einleiten und Risiken oder Schäden abwenden kann.

Anmerkung: Beim Erstellen eines Prozesses ist der EC-Public-Key des Empfängers nicht notwendig. Die Kontaktabfrage für das Hoch- und Herunterladen ist identisch. Beim Herunterladen von Daten ist das jeweilige Zertifikat für den EC-Public-Key des Senders notwendig, um etwaige Signaturen zu validieren.

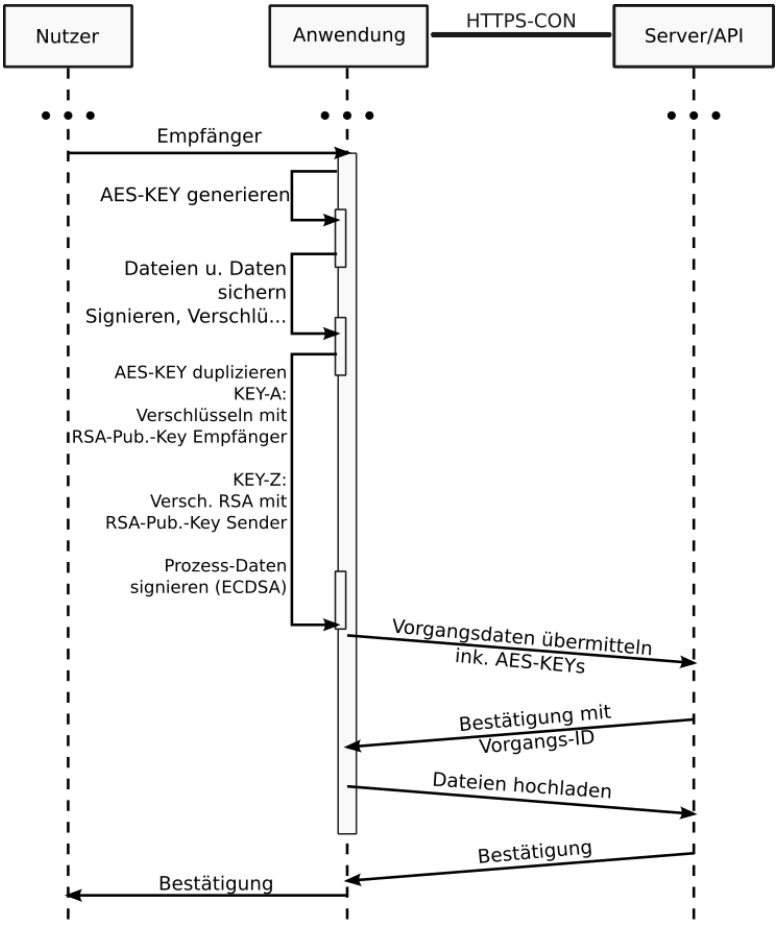

Abbildung 7: Prozesserstellung Ablaufdiagramm

Ist der Empfänger gewählt, so beginnt die Anwendung damit, einem AES-Schlüssel zu generieren. Wie bereits in Abschnitt *5.2.4 Datenlebenszyklus* erläutert, wird dieser Schlüssel im Anschluss genutzt, um die Prozessdaten zu verschlüsseln und zu signieren. Der AES-Schlüssel ist ebenfalls schützenswert. Es ist daher zu verhindern, dass Dritte diesen im Klartext nutzen können, um entsprechende Daten/Dateien auszulesen. Daher wird der AES-Schlüssel zunächst dupliziert. Ein Schlüssel wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Das Duplikat des Schlüssels wird mit dem öffentlichen Schlüssel des Senders verschlüsselt. Durch die Eigenschaften von RSA ist so sichergestellt, dass niemand, außer Sender und Empfänger, die AES-Schlüssel entschlüsseln kann. Zur Entschlüsselung nutzt der jeweilige Client den privaten Schlüssel. Der öffentliche Schlüssel des Empfängers stammt aus dem verifizierten Zertifikat. Vor dem eigentlichen Hochladen der Dateien werden zunächst die Prozessdaten signiert. Das Signieren der Prozessdaten und Dateien geschieht mit dem EC-Schlüsselpaar. Daraufhin werden die Prozessdaten hochgeladen. Erst nach einer darauffolgenden Erfolgsmeldung des Hintergrundsystems werden die zu transportierenden Dateien hochgeladen. Beim Empfangen eines Prozesses wird dem jeweiligen Nutzer entsprechend auch nur der symmetrische Schlüssel übergeben, der für ihn bestimmt ist. Durch das Entschlüsseln des AES-Key mit Hilfe des privaten

Schlüssels kann der Empfänger alle Daten/Dateien entschlüsseln. Das Hochladen neuer Dateien zu einem Prozess geschieht mit demselben AES-Key. Grundsätzlich prüft der Empfänger die Signaturen der Prozessdaten und Dateien auf ihre Korrektheit und stellt so Integrität und Nachweisbarkeit der Daten/Dateien fest.

5.2.6 Fehlerfälle und sichere Datenvernichtung

Grundsätzlich werden im Fehlerfall (Exceptions) Aufzeichnungen/Log-Einträge angefertigt. Diese Einträge beinhalten ausschließlich standardisierte Meldungen, warum der Fehler und wann er aufgetreten ist. Sensible Daten wie Nutzernamen, Patientendaten im Allgemeinen oder Schlüssel für eine Chiffrierung dürfen unter keinen Umständen in die Fehlerbeschreibung mit einfließen.

Sind bei einem Fehlerfall Dateien involviert, bspw. beim Herunterladen einer Auswertung, so ist die Anwendung so konzipiert, entsprechende Dateien im Arbeitsspeicher zu überschreiben. Dieser Fall wird in der BSI Testcharakteristik *O.Source_7* der TR-03161-1 beschrieben. Es wird hier angenommen, dass Dateien in einem Byte-Array (*byte[]*) gelagert werden. Kommt es zu einem Fehler, ist es wichtig, zu verstehen, dass dieses Byte-Array nicht einfach auf *null/NULL* gesetzt werden darf. Denn auch wenn direkt im Anschluss eine Garbage-Collection den Adressraum freigibt, ist nicht garantiert, dass die entsprechenden Bytes auch überschrieben werden. Es reicht daher nicht aus, den Adressraum freizugeben. Er MUSS manuell überschrieben werden.[79]

Eine einfache Lösung könnte die folgende sein:

```
1 //function to override data on exceptions
2 public void dataOverrride(byte[] data) {
3     for (i = 0; i < data.Length; i++)
4         data[i] = 0;
5 }
```

Listing 1: Data override (C#)

6 Analyse

Um das Konzept systematisch zu prüfen, werden zunächst die Prüfaspekte des BSI aus dem Kapitel *4 Prüfaspekte (Tabelle 2)* gegen das im Kapitel *5 Prototyp* beschriebene Konzept gegenübergestellt. Dabei wird argumentiert, ob die entsprechenden Kategorien der zu prüfenden Aspekte abgedeckt sind. Sind einzelne Prüfaspekte nicht berücksichtigt worden, so wird offengelegt, warum dies so ist und welche Prüfaspekte betroffen sind. Das Ergebnis einer Prüfung kann drei Resultate ergeben:

1. **Erfüllt:** Ein Prüfaspekt ist vollständig abgedeckt und damit erfüllt.
2. **Ergebnislos:** Das Ergebnis ist nicht schlüssig.
3. **Verstoß:** Der Prüfaspekt wurde nicht erfüllt, beachtet oder es mangelt an einer Implementierung.

In den Testcharakteristika des BSI gibt es zusätzlich das Resultat *NOT APPLICABLE*. Dies bedeutet: Der Aspekt konnte mangels Implementierung nicht angewendet werden. Hier wurde sich dazu entschieden, dieses Ergebnis mit dem Resultat *Verstoß* zusammenzulegen, da aufgrund des Kapitels *4 Prüfaspekte* bereits alle Aspekte entfallen, welche klar nicht den Anforderungen an das Konzept gerecht werden. Das bedeutet, es werden nur Prüfaspekte geprüft die, laut Anforderung, auch implementiert sein sollen. Prüfaspekte, welche argumentativ sowohl *Verstoß* als auch mit Begründung *Erfüllt* sein können werden dem Ergebnis *Ergebnislos* angerechnet.

Ist der Abgleich der Prüfaspekte durchgeführt, werden alternative Verfahren verargumentiert und aufgezeigt.

6.1 Abgleich der Prüfaspekte

6.1.1 Anwendungszweck

Bezogen auf die personenbezogenen Daten der Praxen ist klar, dass die jeweilige Praxis über die Speicherung und Nutzung der Daten informiert werden muss. Dies geschieht im Rahmen der Nutzungsbedingungen, welche eine Nutzereinwilligung bezüglich dieser Daten enthält. In dieser wird der Nutzer ausdrücklich darauf hingewiesen, welche Daten wie und warum genutzt werden. Ohne Nutzereinwilligung ist eine Verwendung der Anwendung nicht möglich. Daher ist auch ein explizites Widersprechen zur Nutzung bestimmter Daten nicht möglich, jedoch auch nicht nötig. Bezogen auf Patientendaten wird darauf hingewiesen, dass die Praxen einen Patienten ausdrücklich darauf hinweisen müssen, wie seine Daten im Rahmen der Untersuchung mit einem Langzeit-EKG verwendet werden. Vor allem, da ein Auswerter als zusätzliche Partei Zugriff auf die entsprechenden Rohdaten besitzt. Einwilligungen der Patienten zur Verwendung der hier beschriebenen Anwendung liegen jedoch in Verantwortung der jeweiligen Praxis. Das Hintergrundsystem besitzt zu keinem Zeitpunkt Patientendaten in unverschlüsselter Form. Daher findet auch keine Verarbeitung dieser statt. Die Anwendung legt sich zur Last, alle Patientendaten ausschließlich verschlüsselt an das Hintergrundsystem zu übermitteln, um die Lesbarkeit für Dritte unmöglich zu machen. Die in den Kapiteln *5.1.3 Einstellungen (Tabelle 3)* und *5.2.3 Prozessablauf und Übertragung* beschriebenen Patientendaten werden, verschlüsselt.

Durch die Anforderungen geht hervor, dass Einwilligungserklärungen durch den Nutzer bzw. die

Praxis jederzeit zurückgezogen werden können. Zugleich kann ein Nutzer seine Einwilligungserklärungen jederzeit digital betrachten. Damit sind die Prüfaspekte O.Purp_06 und O.Purp_07 abgedeckt.

Der Prüfaspekt Purp_08 bezieht sich unter spezifischer Betrachtung auf die Weitergabe von Daten an Drittanbieter. In diesem Zusammenhang wird dieser Aspekt jedoch allgemein betrachtet, da Drittanbieterpakete ausgeschlossen sind. Das bedeutet, dass es keinem Dritten, sowohl potenziellen Angreifer als auch Nutzern, die einem Prozess nicht zugeordnet sind, nicht möglich sein soll, sensible Daten zu erhalten. Die in den kommenden Abschnitten nachfolgenden Prüfaspekte prüfen einen solchen Schutz durch die Anwendung. Deshalb wird dieser Punkt hier als erfüllt betrachtet. Sensible Daten die befugten Nutzern angezeigt werden, werden im kleinstmöglichen Rahmen gehalten. Daher werden Patientendaten in unverschlüsselter Form ausschließlich dazu genutzt, einen Patienten für den Nutzer identifizierbar zu machen. Bei der Betrachtung eines Prozesses wird zudem auch der Praxisname des jeweiligen Senders angezeigt, damit dieser für den Empfänger eindeutig identifiziert werden kann. Dies sind nur die nötigsten Daten, um einen Vorgang zu identifizieren.

6.1.2 Architektur

Bei der Planung und Umsetzung der Architektur muss die Sicherheit eine übergeordnete Rolle spielen. Es ist zu erkennen, dass in der Phase der Planung bereits die Betrachtung der Thematik eine hohe Priorität hat. Es ist von vornherein klar, dass diese Anwendung sensible Daten transportieren soll und entsprechend darauf konzeptioniert wird. Erkennbar wird dies auch am abgebildeten Datenlebenszyklus (Abb. 5). Hier ist jedoch anzumerken, dass das lokale Speichern der Dateien in der Verantwortung der jeweiligen Praxis liegt. Lediglich das Hintergrundsystem ist, unter Betrachtung des Gesamtsystems, berechtigt, Dateien langfristig aufzubewahren. Der Prozesszyklus aus Abb. 3 und 4 dient gleichzeitig als Lebenszyklus für Schlüsselmaterial. Denn existiert ein Prozess nicht mehr, so existieren auch dessen Schlüssel nicht mehr. Der Lebenszyklus der asymmetrischen Schlüsselpaare ist offensichtlich der Anwendungsbeschreibung entnehmbar. Schlüsselpaare bleiben solange, bis eine bewusste Löschung des Containers eintritt oder eine Deinstallation stattfindet (mehr zum kryptographischem Material 6.1.9).

Vorgehensweisen zur Datensicherung sind im Bezug auf lokale Backups in der Verantwortung der Praxis. Das Sichern bspw. des Containers ist unverschlüsselt nicht möglich. Eine unverschlüsselte Form des Containers existiert nur im Arbeitsspeicher nach erfolgreicher Eingabe des Passworts. Datensicherungen des Hintergrundsystems sind aufgrund der Tatsache, dass lediglich verschlüsselte sensible Daten durch die Anwendung übermittelt werden, zunächst einmal geschützt. Eine detailliertere Betrachtung des Hintergrundsystems bezüglich der Backupverfahren findet aufgrund des Fokus dieser Arbeit hier nicht statt. Daher gilt der Prüfaspekt O.Arch_4 hier als erfüllt.

Das Hintergrundsystem besitzt allein für die Authentifizierung eines Nutzers, Sicherheitsfunktionen. Zusätzlich ist eine Signaturvalidierung beim Hochladen von Dateien ersichtlich. Das bedeutet: Sowohl der Client als auch das Hintergrundsystem kann Signaturen validieren und damit Integrität und Nachweisbarkeit prüfen. Ein Eingriff in Form einer Entschlüsselung ist seitens des Hintergrundsystems dafür nicht notwendig. Damit besitzen alle hier erkennbaren API-Endpunkte notwendige Sicherheitsfunktionen. Daher ist der Prüfaspekts O.Arch_5 als erfüllt betrachtet.

6.1.3 Quellcode

Nutzereingaben (z.B. manuelle Eingabe von Patientendaten) sind grundsätzlich Escaped und stellen ein geringes Risiko da. Wird eine GDT-Datei genutzt, so wird diese validiert und bei auftretenden Ungereimtheiten verworfen. Andere Dateien werden jedoch durch das empfohlene Vorgehen der C# Dokumentation über einen FileStream in den Speicher geladen. Das BSI gibt zum Prüfaspekt O.Source_1 jedoch nicht genau an, was als vertrauenswürdige Quelle zählt und was nicht.[13, p.33] Daher ist dieser Prüfaspekt ergebnislos. In Bezug auf Fehler (Exceptions) wird davon ausgegangen, dass alle Fehler abgefangen, behandelt und dokumentiert werden. Ein daraus resultierender Log-Eintrag in einem Log-File enthält zu keinem Zeitpunkt sensible Daten. Weder Daten des Anwenders, noch eines Patienten. Lediglich die Fehlermeldung, dessen Datum und mögliche Ursachen werden aufgelistet. Bei einem Fehlerfall im Zusammenhang mit Dateien werden IMMER die bereits eingelesenen Bytes überschrieben und der Vorgang abgebrochen. Mögliche Fehlerfälle sind aus Abb. 5 zu entnehmen.

C# hat zwar die Möglichkeit, über Zeiger manuell Speicheradressen zu verwalten[80], jedoch ist dies eher unüblich, da eine automatische Speicherverwaltung (Garbage collection) vorhanden ist.[79] Es ist daher nötig, den Prüfaspekt O.Source_6 abzugleichen, jedoch nur um festzustellen, ob eine manuelle Speicherverwaltung stattfindet. Dies ist hier nicht der Fall. Es ist insoweit wichtig, da O.Source_7 eine sichere Datenvernichtung voraussetzt. Wird bspw. eine Datei in den Speicher geladen und soll im Anschluss vernichtet werden, so reicht es nicht aus das entsprechende Byte-Array gleich NULL zu setzten und die Garbage collection (CG) auszulösen. Wird die CG nicht manuell ausgelöst, ist der Zeitpunkt einer CG nicht garantiert und der Speicher enthält weiterhin die Bytes. Auch ist nach einer CG nicht zu 100% garantiert, dass die Bytes überschrieben werden oder schlicht der Speicher nur freigegeben wird.[79] Daher muss das im Kapitel *5.2.6 Fehlerfälle und sichere Datenvernichtung* beschriebene Verfahren IMMER angewendet werden wenn Daten nicht mehr benötigt werden. Damit ist der Aspekt O.Source_7 erfüllt.

6.1.4 Authentifizierung

Bei der lokalen Authentifizierung (freigeben des Containers) wird nicht speziell das Ausprobieren des Passworts verhindert. Jedoch existieren Maßnahmen, die diesen Prozess deutlich verlängern. Diese sind durch die Eigenschaften von PBKDF2 gegeben. Wie bereits im vorangegangenem Abschnitt 4.9.9 verargumentiert wurde, befindet sich PBKDF2 im Rahmen der Prüfaspekte. Es sollte jedoch erneut evaluiert werden, ob Argon2 eine zulässige Alternative darstellt, sofern eine entsprechende Implementierung in .NET nachgeliefert wird. Ist eine Anmeldung lokal, als auch gegenüber des Hintergrundsystems einmal durchgeführt, besteht diese Verbindung während der aktiven Nutzung. Wird jedoch länger als 10 Minuten keine Aktivität durchgeführt (Idel-Modus) trennen Client und Server die Verbindung und eine erneute Authentifizierung ist erforderlich.

Das Ändern von Authentifizierungsdaten (Schlüsselpaar) ist in dem beschriebenen Konzept nur in Zusammenhang mit dem Betreiber möglich, da entsprechende Zertifikate, die auf dem öffentlichen Schlüssel basieren, erneuert werden müssen. Das Erneuern des Containers entzieht ebenfalls den Zugriff auf bestehende Prozesse, da ein neues Schlüsselpaar erzeugt wird. Ein Ändern des Passworts für die lokale Authentifizierung ist hier jedoch, soweit beschrieben, möglich. Da die Änderung der Authentifizierungsmittel ohne weiteres nicht erfolgen kann, ist hier kein Bruch des Prüfaspektes O.Auth_10 zu erkennen.

6.1.5 Passwörter

Beim Festlegen eines Passworts wird der Nutzer laut Konzept darüber informiert, wie stark dieses Passwort ist. In diesem Rahmen wird davon ausgegangen, dass diese Prüfung keinen Rückschluss auf das tatsächliche Passwort zulässt. Eine Quellcodeanalyse kann hier nicht durchgeführt werden. Wie bereits in vorangegangenen Abschnitten beschrieben, ist es möglich, das Passwort für den Container zu ändern. Dabei ist es jedoch nicht möglich, aber auch nicht nötig, den Nutzer darüber zu informieren. Der Prüfaspekt O.Pass_4 geht davon aus, dass es sich hier um ein Passwort für eine "klassische" Anmeldung handelt. Jedoch ist der Nutzer in diesem Fall lokal für sein Passwort zuständig. Ein Ändern des Passworts würde das alte Passwort voraussetzen und entsprechend der fehlenden Verfügbarkeit des Containers auffallen. Daher ist ein Informieren über eine eventuelle Änderung des Passworts so nicht nötig.

Da das Passwort für eine Passwortbasierte Schlüsselableitung genutzt wird, ist ein Speichern des Passworts nicht notwendig. Daher werden auch keine in O.Pass_5 verlangten Hash-Algorithmen benötigt. Die Verfahren für die Passwortbasierte Schlüsselableitung sind dem Abschnitt *5.1.4 Kryptografische Algorithmen* zu entnehmen.

6.1.6 Sitzungen

Das für die Sitzung gültige Token wird durch das Hintergrundsystem erzeugt. Der Client ist dafür verantwortlich, das Token als sensible Information zu behandeln. Entsprechend wird das Token zu keinem Zeitpunkt permanent gespeichert, sondern nur im flüchtigen Speicher verwahrt. Es ist festzustellen, dass ein Token wie ein Chiffrierschlüssel in einer sicheren Umgebung aufzubewahren ist.

Es kann immer nur eine Sitzung gleichzeitig von einer Praxis ausgehen. Das Hintergrundsystem kann ausschließlich eine Sitzung pro Nutzer handhaben. Es ist daher nicht nötig spezifische Token zu sperren. Ein Abmelden oder Beenden der Anwendung vernichtet, ein entsprechendes Token zu diesem Nutzer lokal als auch extern auf dem Hintergrundsystem.

6.1.7 Datensicherheit

Es gibt nur bedingt Einstellungen, die eine Auswirkung auf die Sicherheit haben. Die Standardeinstellungen sind die des 'GDT-ONLY'-Modus. Dieser wird im Vergleich zur manuellen Eingabe von Patientendaten als sicherer betrachtet. Grund dafür ist primär eine geringere Fehlerwahrscheinlichkeit durch den Faktor Mensch.

Sensible Daten, in diesem Kontext Prozessdaten und Dateien, werden nur wenn nötig entschlüsselt. Prozessdaten zur Betrachtung von Prozessen. Dateien "*just-in-time*"bei einer lokalen Ablage. Die Anwendung ist in der Verantwortung, entsprechende Daten ausschließlich verschlüsselt an das Hintergrundsystem zu übermitteln. Verschlüsselte Daten befinden sich daher ausschließlich auf dem Hintergrundsystem, ausgenommen des Containers, welcher sich durch seinen Zweck auf der Clientseite befinden muss. Sensible Daten werden ausschließlich für ihren Zweck entschlüsselt (O.Data_2 und O.Data_7 erfüllt). Das im Rahmen dieser Arbeit beschriebene Konzept verwendet keine geteilten Speicherbereiche, Dienste oder Interfaces, wie sie in O.Data_4 genannt werden.

Die in O.Data_5 beschriebenen Anforderungen zur Vernichtung von Daten nach ihrer Verwendung ist bereits in Abschnitt *6.1.3 Quellcode* beschrieben. Diese Vernichtung gilt auch beim Beenden der Anwendung. Daher gilt an dieser Stelle dieser Prüfaspekt als erfüllt.

Bei einer Gegenüberstellung der verwendeten Daten und deren Zweck ist erkennbar, dass ausschließlich Daten erhoben werden, die auch tatsächlich ihren Nutzen erfüllen. Bspw. werden Patientendaten nur genutzt, um einen Prozess besser zuordnen zu können. Unter der Erhebung von Daten wird hier ausschließlich das Verwenden oder Auslesen aus Dateien im Sinne des Transports eines Prozesses verstanden. Ein Auswerten der Daten z.B. für Statistiken ist nicht vorgesehen und erfolgt auch nicht. Die Datensparsamkeit und Zweckbindung ist daher gegeben.

Die Anwendung bietet keine Funktion, den privaten Schlüssel oder anderes Schlüsselmaterial zu exportieren. Generell können keine Daten per Funktion aus der Anwendung exportiert werden. Lediglich der Container in verschlüsselter Form kann aus den Anwendungsdaten kopiert werden. Dies ist jedoch eine gewollte Eigenschaft zur Verwendung desselben Containers an mehreren Rechnern einer Praxis. Bei der Deinstallation der Anwendung wird dieser Container vernichtet. Ein Authentifizieren gegenüber dem Hintergrundsystem ist daher nicht mehr möglich. Unabhängig der Praxis kann der Betreiber durch das Entfernen des öffentlichen Schlüssels im System einen Zugang abstellen. Eine Authentifizierung dieses speziellen Clients ist dann nicht mehr möglich.

6.1.8 Netzwerkkommunikation

Das Serverzertifikat wird bei jedem Start validiert. Ohne dieses Zertifikat ist es der Anwendung nicht möglich, Zertifikate anderer Nutzer zu verifizieren. Zusätzlich kann es genutzt werden, um generell Antworten des Hintergrundsystem zu validieren. Dies geschieht jedoch nur bei der Übermittlung anderer Nutzer. Laut O.Ntwk_6 ist es notwendig, jede Antwort des Hintergrundsystems auf seine Authentizität und Integrität zu prüfen. Das geschieht hier mit Hilfe der durch TLS 1.3 gegebenen Eigenschaften. Da die Kommunikation mit dem Hintergrundsystem ausschließlich über TLS 1.3 stattfindet, ist die Authentizität und Integrität aller Nachrichten gewährleistet. Dazu nutzt TLS 1.3 eine "Authenticated Encryption with Associated Data" (AEAD).[81, 49]

6.1.9 Kryptografische Umsetzung

Schlüsselmaterial darf laut den Prüfaspekten für kryptografische Umsetzung der TR-03161-1 des BSI nicht fest einprogrammierte sein. Ausgenommen ist Schlüsselmaterial mit entsprechendem Schutz gegen "reverse engineering".[13, p.37] Das in dieser Anwendung verwendete Schlüsselpaar gilt nicht als fest einprogrammiert. Es unterscheidet sich von Anwender zu Anwender, da eine Generierung dieser Schlüssel bei der Einrichtung der Anwendung stattfindet. Für das Schlüsselmaterial und die kryptografischen Verfahren ist die TR-02102-1[33] zu beachten. Für Zertifikate gilt die TR-02102-2[33] und TR-2103[34]. Für die Verbindung zwischen Client und Server wird davon ausgegangen, dass die TR-02102-2[33] beachtet wurde und TLS in der aktuellsten Version "sauber" eingesetzt wird. Die Prüfaspekte O.Cryp_2 und O.Arch_3 sind damit hier als erfüllt betrachtet. Für den Austausch der öffentlichen Schlüssel werden x509v3 Zertifikate eingesetzt, welche der TR-2103[34] entsprechen müssen. Da der Aussteller der Zertifikate in diesem Zusammenhang immer als vertrauenswürdig betrachtet wird, gilt diese TR als beachtet.

Die verwendeten Verfahren aus *Tabelle 4 Angewendete Verfahren in Übersicht und Funktion* sind gegen die TR-02102-1[33] abzugleichen. Hierbei ist zu erkennen, dass alle Schlüssel eine geeignete Stärke nach O.Cryp_5 aufweisen.

Nach O.Cryp_4 darf ein Schlüssel nur einen Zweck erfüllen. In Bezug auf die symmetrischen Schlüssel ist es das Schützen der Daten und Dateien während der Übertragung und Verwahrung. Es kann jedoch argumentiert werden, dass ein Schlüssel mehr als einen Zweck erfüllt, wenn er für den Transport oder Lagerung unterschiedlicher Dateien verwendet wird. Generiert der Zuweiser einen Schlüssel, um Dateien zu übertragen, so wird dieser auch genutzt, um die entsprechende Auswertung dieser Dateien zu schützen. Sprich: Ein Schlüssel wird sowohl für das Verschlüsseln der Rohdaten als auch der ausgewerteten Daten genutzt. Dem ist entgegenzusetzen, dass der jeweilige Schlüssel nicht nur für den Schutz der Dateien genutzt wird. Er ist für den gesamten Schutz eines Prozessdurchlaufs in Verwendung. Daher ist hier O.Cryp_4 als erfüllt zu betrachten.

Auch bei asymmetrischen Schlüsseln ist der Prüfaspekt O.Cryp_4 erfüllt. Eine Authentisierung (Signatur) wird durch ein EC-Schlüsselpaar gesichert. Die eigentliche Ver- und Entschlüsselung der symmetrischen Schlüssel wird durch das RSA-Schlüsselpaar gesichert. Sprich, der Schlüssel für die **Authentisierung** ist nicht mit dem Schlüssel für den **Schutz** identisch.

Symmetrische Verschlüsselung:
Für die symmetrische Verschlüsselung wird AES mit einer Schlüssellänge von 256 Bit im Modus CBC eingesetzt. Dies ist nach der TR-02102-1 eines der empfohlenen Verfahren.[33, p.27] Die Schlüssellänge muss mindestens 128 Bit betragen. Um einen maximalen Schutz zu gewährleisten, wird hier auf die höchste Schlüssellänge zurückgegriffen. Das angewendete Verfahren als auch dessen Schlüssellänge entsprechen den Empfehlungen des BSI.

Signaturen:
Für die Anfertigung der Signaturen wird ECDSA mit einem entsprechendem EC-Schlüssel eingesetzt. Mit einer Länge von 250 Bit befindet sich der Schlüssel in einer geeigneten Länge. Das Verfahren ECDSA lässt sich aus der TR entnehmen. Aufgrund der Eigenschaften von ECDSA ist es anders als bei RSA hier nicht notwendig, weitere Schemen anzuwenden bzw. zu beschreiben.

Asymmetrische Ver- u. Entschlüsselung:
Für die Ver- und Entschlüsselung der symmetrischen Schlüssel wird RSA mit EME-OAEP-Padding bzw. Schema verwendet. Die Verwendung von EME-OAEP im Zusammenhang mit asymmetrischer Ver- und Entschlüsselung mit RSA ist die einzige empfohlene Variante. Eine Anwendung des PKCS#1v1.5-Padding ist nach der vorangegangenen Argumentation weder empfehlenswert, noch zugelassen. (Siehe *4.9.6 Asymmetrische Ver- und Entschlüsselung*)

Passwortbasierte Schlüsselableitung:
Eine Passwortbasierte Schlüsselableitung mit - wie hier verwendet - PBKDF2 ist mit 10.000 Iterationen nach BSI ausreichend. Unter Berücksichtigung, dass der Container, welcher mit dem abgeleiteten Schlüssel chiffriert wird, als Datei leicht zu verschieben ist, sollte bei der Weiterentwicklung, über den Prototypen-Status hinaus, eine größere Iterationsanzahl und eine Verwendung von hardwareseitigen Schutzmaßnahmen evaluiert werden. Jedoch ist im Rahmen dieser Arbeit und unter Berücksichtigung der gegebenen Vorgaben durch Richtlinien und dessen Einschränkungen hier kein Mangel festzustellen.

CSPRNG
Um Schlüsselmaterial zu erzeugen, nutzt die Anwendung die Standardfunktion der Sprache C# (.NET). Nach vorangegangener Begründung aus *4.9.10 Cryptographically Secure Pseudorandom Number Generator (CSPRNG)* werden die hier verwendeten Standardfunktionen als sicher betrachtet. Das setzt jedoch die Anforderung, dass die Anwendung auf einem FIPS 140-2-Zertifizierten Betriebssystem laufen muss. Sprich: Eine aktuelle Windows Version (mindestens Windows 7).[67, 68] Im Zusammenhang mit diesen Anforderungen werden damit die Prüfaspekte O.Rand_1 bis O.Rand_4 als bestanden angesehen.

Anmerkung: Im Zusammenhang mit dem Prüfaspekt O.Rand_1 wird auf die TR-03116 des BSI verwiesen. Diese empfiehlt für die Erzeugung von Zufallszahlen unter anderem die Verwendung eines NTG.1.. Solche werden speziell in Kapitel *4.9.10 Cryptographically Secure Pseudorandom Number Generator (CSPRNG)* genauer betrachtet.

Übersicht:

	Prüfaspekt	
Kategorie	**Erfüllt**	**Ergebnislos o. Verstoß**
Anwendungszweck	O.Purp_01 - O.Purp_09	
Architektur	O.Arch_01 - O.Arch_05	
Quellcode	O.Source_2 - O.Source_7	O.Source_1 (Ergebnislos)
Authentifizierung	O.Auth_6, O.Auth_8 - O.Auth_11	
Passwörter	O.Pass_01 - O.Pass_05	
Sitzungen	O.Sess_2 - O.Sess_6)	
Datensicherheit	O.Data_01, O.Data_02 O.Data_04 - O.Data_07 O.Data_12, O.Data_15 O.Data_17, O.Data_18	
Netzwerkkommunikation	O.Ntwk_6, O.Ntwk_5	
Kryptographische Umsetzung	O.Cryp_01 - O.Cryp_05 O.Rand_1 - O.Rand_4	

Tabelle 8: Ergebnislose und nicht bestandene Prüfaspekte

6.2 Alternativen

Die Analyse (*6.1.9*) zeigt, dass die beschriebenen Anforderung bzw. die Umsetzung der Anwendung den Prüfaspekten der BSI Richtlinien entsprechen. Damit ist der vom BSI vorgegebene Mindeststandard erfüllt. Im Folgenden werden alternative kryptografische Umsetzungen erläutert und gegen die im Kapitel *5 Prototyp* beschriebenen Umsetzungen verargumentiert. Dabei gilt der Grundsatz, dass alle hier beschriebenen Alternativen ebenfalls den Standard der BSI-Prüfaspekte erfüllen. Ein erneutes Prüfen der Testcharakteristika ist daher nicht notwendig.

6.2.1 Prozessverschlüsselung mit AEAD

Im folgenden Abschnitt soll die Prozessdateienverschlüsselung mit einer "Authenticated Encryption with Associated Data" (AEAD) ergänzt werden. Dabei sind folgende Anforderungen zu beachten:

(A) Die Integrität und Nachweisbarkeit muss ohne Entschlüsseln verifizierbar sein.

(B) Prozessdaten und -dateien müssen getrennt voneinander auf Integrität und Nachweisbarkeit prüfbar sein.

(C) Prozessdaten und -dateien müssen verschlüsselt sein.

(D) Prozessdaten müssen unabhängig der Prozessdateien entschlüsselt werden können.

Zunächst ist festzustellen, ob die Prozessdaten unabhängig der Dateien zu entschlüsseln sind. Das geschieht für die Vorschau der Vorgänge/Prozesse. Es kommen also zwei unterschiedliche Verschlüsselungsverfahren zum Einsatz. Ein "klassisches" Verfahren (CBC) und ein AEAD-Verfahren. Durch den Ablauf beim Herunterladen von Dateien ist sicher, dass vor dem Herunterladen die Prozessdaten bereits vorliegen und entschlüsselt sind. Es folgt die Überlegung, die Prozessdaten als verbundene Daten (AD) für eine AEAD der Prozessdateien zu verwenden.

Es ist zunächst die Initialisierung eines neuen Prozesses zu betrachten. Bei der Initialisierung werden zunächst die Prozessdaten (Siehe Tabelle 6) zusammengetragen. Hier ist festzustellen, welche Prozessdaten als zusätzliche Daten infrage kommen. Bspw. ist das Datum der letzten Änderung schlecht geeignet, da solche Werte vorzugsweise durch das Hintergrundsystem verwaltet werden. Ähnliches gilt auch für den Status. Es kommen also lediglich Daten infrage, die in der Kontrolle des Clients liegen und/oder keine Veränderung im Verlaufe der Prozessabarbeitung erfahren (bspw. die Vorgangsnummer [Prozess-ID]). Es kommen demzufolge folgende Daten infrage: *Vorgangsnummer*, *Patientendaten*, *Praxisname (Sender und Empfänger)*, *Erstellungsdatum* und *AES-IV*.

Anmerkung: In diesem Fall wird der Praxisname und nicht der Identifikator verwendet, da die Verschlüsselung die Aufgabe des Clients ist. Nach Anforderung darf kein Nutzer den Ident eines anderen Nutzers erfahren. Daher müssen hier die Praxisnamen verwendet werden.

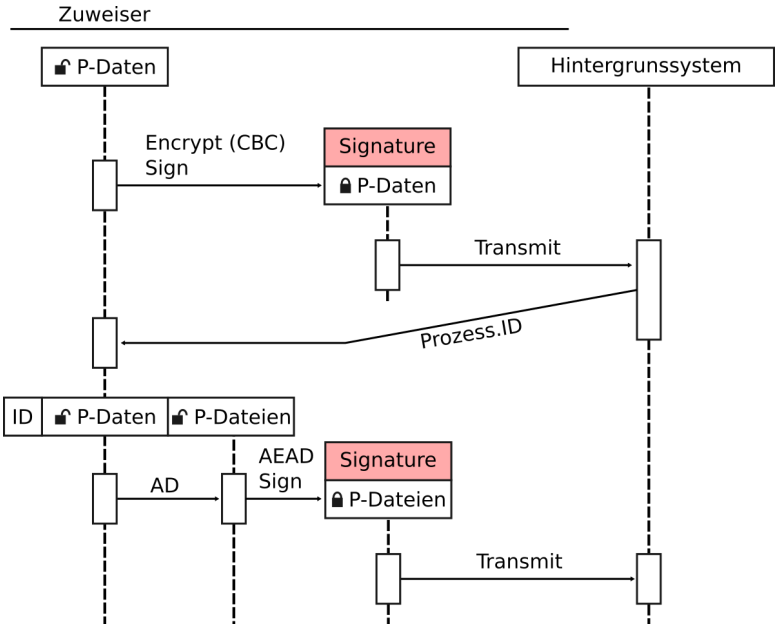

Abbildung 8: Alternative mit AEAD Zuweiser (Versimpelung)

Um diese als AD zu verwenden, muss zunächst die Prozess-ID durch das Hintergrundsystem gewonnen werden. Dafür verschlüsselt und signiert die Anwendung die Prozessdaten und übergibt sie dem Hintergrundsystem. Die Anwendung überträgt hier die verschlüsselten Daten, behält jedoch die selbigen unverschlüsselt. Die API antwortet bei erfolgreicher Übertragung mit einer entsprechenden Prozess-ID. Diese wird infolgedessen den, in der Anwendung unverschlüsselten, Prozessdaten angehängt. Die Prozessdateien können dann mit den unverschlüsselten Prozessdaten als AD chiffriert werden. Auf die Chiffrierung folgt auch bei den Dateien eine Signatur. Dateien inkl. Signatur werden folglich an das Hintergrundsystem übermittelt.

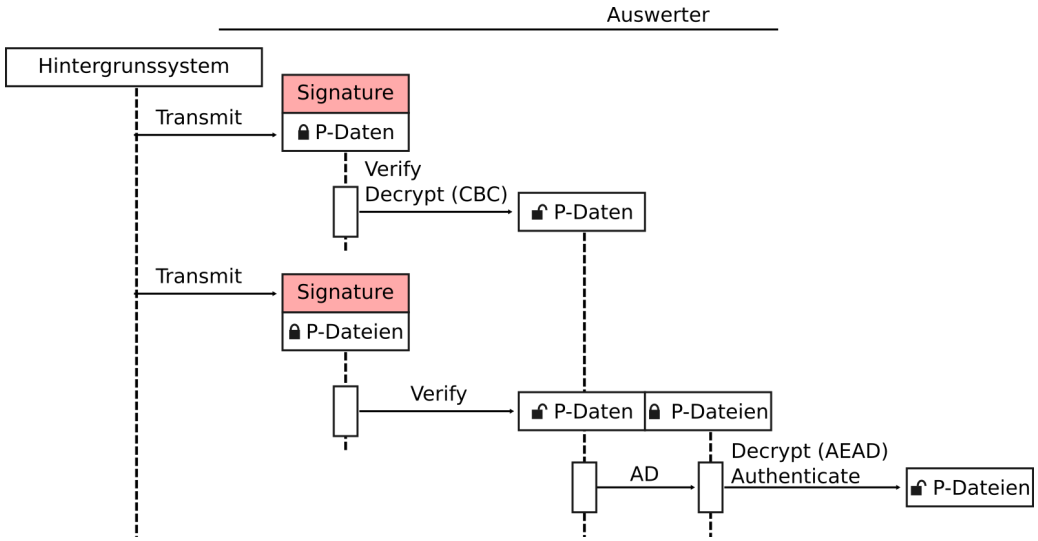

Abbildung 9: Alternative mit AEAD Auswerter (Versimpelung)

Um Dateien zu empfangen, muss der Nutzer gezielt einen Prozess auswählen, von dem er die Dateien erhalten möchte. Dem zufolge sind die Prozessdaten bereits vorhanden, validiert sowie entschlüsselt. Fordert der Nutzer nun die Dateien zu einem Prozess an, so werden diese

Heruntergeladen und verifiziert/validiert. Die bereits vorhandenen Prozessdaten werden nun, wie bereits bei der Verschlüsselung, als AD genutzt, um die Dateien zu entschlüsseln.

Abwägung
Der Einsatz einer Verschlüsselung mit zusätzlichen Daten bietet nur bedingt Vorteile. Durch eine AEAD wird sichergestellt, dass die richtigen Dateien auch grundsätzlich mit den richtigen Daten verwendet werden. Eine getrennte Verwendung ist durch die Eigenschaften von AEAD ausgeschlossen. Werden Prozessdaten erst nach dem Verschlüsseln signiert, so kann ein Empfänger nur verschlüsselte Prozessdaten verifizieren. Unter Anwendung von AEAD kann auch der Klartext nach dem Entschlüsseln in Kombination mit den Prozessdateien auf Authentizität und Integrität geprüft werden. Dies kann ein erhöhter Schutz sein. Jedoch kann dieser auch durch eine zusätzliche Signatur erreicht werden. Um letzteren Punkt zu erreichen, muss jedoch AEAD in der Betriebsart CCM verwendet werden, da nur CCM nach "Mac-then-Encrypt" arbeitet. Eine umgekehrte Herangehensweise wie bei GCM ("Encrypt-then-Mac") bringt diese Vorteile nicht, da die Verifizierung des verschlüsselten Anteils bereits durch Signaturen gesichert ist. Für einen Prototypen ist der erhöhte Implementierungsaufwand und die nur geringen Vorteile nicht zweckdienlich. Daher wird nur bei einer Weiterentwicklung dieses Vorgehen zur Auswahl gestellt.

6.2.2 Authentifizierung mit Mutual Transport Layer Security (mTLS)

Mutual Transport Layer Security (mTLS) ist wie die SSH-Public-Key-Authentication eine mutual authetifikation und stellt sicher, dass der zu authentifizierende Teilnehmer im Besitzt des erwarteten privaten Schlüssels ist. mTLS ist in der hier verwendeten TLS-Version 1.3 vorhanden und ist eine erweiterte Sicherheitsmaßnahme für einen sicheren Verbindungsaufbau. Bei einem üblichen TLS-Handshake bekommt der Client nach der, von ihm ausgehenden, Initialisierungsnachricht (Client-Hello) ein Zertifikat des Servers. Mit diesem kann der Client prüfen, ob es sich bei dem Verbindungspartner tatsächlich um den erwarteten Teilnehmer handelt. Darauf folgt eine Schlüsselvereinbarung und ein verschlüsselter Kanal wird aufgebaut. Bei mTLS wird dieser Handshake um die Authentifizierung des Clients erweitert. Nachdem das Zertifikat des Servers durch den Client als authentisch eingestuft ist, übergibt der Client dem Server sein Zertifikat und eine Zertifikatsvalidierung (Certificate Verify). Die Zertifikatsvalidierung ist dabei die Lösung der Challenge als signierter Hashwert. Die Challenge setzt sich aus dem gesamten Kommunikationsverlauf zusammen und ist damit entsprechend einzigartig. Ist das übergebene Zertifikat valide und die Challenge kann reproduziert und damit die Signatur verifiziert werden, dann ist die Authentifizierung erfolgreich und eine Schlüsselvereinbarung findet statt.[81, 11, 10]

Anmerkung: Es handelt sich hier um eine simplifizierende Darstellung des mTLS Handshake, um den eigentlichen Kontext zu umreißen. Für eine detaillierte Beschreibung sei an dieser Stelle auf den Standard [RFC-8446][81] und entsprechende Fachliteratur [11, 10] verwiesen.

Abwägung
mTLS ist ein bereits etabliertes Verfahren und ist daher bereits in realen Anwendungen geprüft. Ein eigens entwickeltes Authentifizierungsverfahren birgt entsprechend der neuen Entwicklung gewisse Risiken, auch wenn es auf anderen Verfahren basiert. Das im Kapitel *5.2.2 Authentifizierungsprozess* beschriebene Verfahren greift jedoch auf Methoden zurück, die bereits durch die Anforderungen in der Anwendung gegeben sind. Der Implementierungsaufwand begrenzt sich daher lediglich auf Nutzung dieser Methoden. Bei mTLS ist daher ein entsprechender Implementierungsaufwand um ein wesentliches höher. Zusätzlich ist es mit mTLS nicht möglich, den Besitz beider asymmetrischen privaten Schlüssel nachzuweisen. Lediglich der Besitz eines privaten Schlüssels kann nachgewiesen werden. Das im Kapitel *5.2.2 Authentifizierungsprozess* beschriebene Verfahren hat das Ziel den Besitz beider privaten Schlüssel nachzuweisen. In der Entwicklung des Prototyps wird daher diese Implementierung bevorzugt und mit dem Fokus auf dessen Sicherheit weiterentwickelt.

7 Fazit

Anhand der, durch die deutschen Behörden, offen liegenden Informationen konnte eine Einordnung der Anwendung als Praxisausstattung durchgeführt werden. Durch diese Einordnung wurde zwischen den drei primären Prüforganisationen, *Technischer Überwachungsverein (TÜV)*, *Bundesinstitut für Arzneimittel und Medizinprodukte (BfArM)* und dem *Bundesamt für Sicherheit in der Informationstechnik (BSI)*, das BSI als zuständige Prüforganisation identifiziert. Als Prüfmittel wurden die, durch das BSI, veröffentlichten technischen Richtlinien [TR-03161-1], [TR-2103], [TR-02102-1] und [TR-02102-2] gewählt. Mit dem, in dieser Arbeit beschriebenen Rahmen, wurden die Prüfaspekte dieser technischen Richtlinien sortiert und eingeschränkt. Diese boten den Leitfaden für die zu entwickelnde Grundlage für eine Anwendung zur Lösung der sicheren Zuweiser-Auswerter-Kommunikation in Auswertegemeinschaften.

Die in dieser Arbeit erzeugte Grundlage für eine Anwendung im Gesundheitswesen entspricht den, für diesen Umfang bestimmten, eingeschränkten Prüfaspekten der zuständigen Behörde. Daher kann das beschriebene Konzept eine Grundlage bzw. einen Ausgangspunkt für eine tatsächlich zum Einsatz kommenden Anwendung bieten. Es ist jedoch anzumerken, dass bei einer Weiterentwicklung aufgrund der starken Einschränkungen bezüglich der Prüfaspekte, eine erweiterte Betrachtung der Prüfaspekte zwingend erforderlich ist. Zusätzlich ist ein immer wieder erneutes Anlegen der Prüfaspekte bei einer Weiterentwicklung dringen notwendig. Es ist angesichts der Alternativen erkenntlich, dass die erarbeitete Grundlage keineswegs der einzige Lösungsweg für eine Lösung der sicheren Zuweiser-Auswerter-Kommunikation ist. Sie kann jedoch als Grundlage fundieren und damit als Ausgangspunkt verwendet werden. Auch wenn die angegebenen kryptografischem Methoden Variation zulassen, so kann jedoch der Ablauf einer Prozessabarbeitung, aufgrund der Eigenschaften der Zuweiser-Auswerter-Kommunikation, nahezu vollständig übernommen werden. Abschließen ist jedoch festzustellen, dass in dieser Arbeit keine vollständige Prüfung, wie sie das BSI durchführen würde, gegeben ist. Daher ist auch nach abschließender Analyse der genutzten Prüfaspekte nicht garantiert, dass eine solche Prüfung in der realen Welt erfolgreich ist.

Bei einer im Verlaufe der Weiterentwicklung folgenden Erweiterung der genutzten Prüfaspekte ist besonders im Hinblick auf die, in dieser Arbeit erzeugte, Dokumentation darauf zu achten, den Umfang der kryptografischen Umsetzung zu erweitern und zu präzisieren. Besonders in Bezug auf eine *Public-Key-Infrastructure* bzw. die Zertifikatsverwaltung und entsprechende Hardwareschutzmaßnahmen. Speziell bei der Betrachtung des Authentifizierungsprozesses sollte eine Verwendung eines etablierten Verfahrens, wie bspw. mTLS, evaluiert werden, sofern eine mutual authetifikation stattfindet.

Abbildungsverzeichnis

Literatur

[1] (2022) Bundesamt für Sicherheit in der Informationstechnik Homepage.
Last Accessed: 02.08.2022. [Online]. Available: https://www.bsi.bund.de

[2] (2022) Bundesinstitut für Arzneimittel und Medizinprodukte Homepage.
Last Accessed: 02.08.2022. [Online]. Available: https://www.bfarm.de

[3] (2018) Datenschutz-Grundverordnung (DSGVO).
Last Accessed: 24.07.2022. [Online]. Available: https://dsgvo-gesetz.de/

[4] (2022) Das Fast-Track-Verfahren für digitale Gesundheitsanwendungen (DiGA) nach § 139e SGB V.
Last Accessed: 02.08.2022. [Online]. Available: https://www.bfarm.de/SharedDocs/Downloads/DE/Medizinprodukte/diga_leitfaden.html?nn=597198

[5] (2022) Digitale Gesundheits- und Pflegeanwendungen.
Last Accessed: 24.07.2022. [Online]. Available: https://www.bfarm.de/DE/Medizinprodukte/Aufgaben/DiGA-und-DiPA/_node.html

[6] (2022) E-Health.
Last Accessed: 24.07.2022. [Online]. Available: https://www.bundesgesundheitsministerium.de/service/begriffe-von-a-z/e/e-health.html

[7] (2021) Die elektronische Patientenakte (ePA).
Last Accessed: 24.07.2022. [Online]. Available: https://www.bundesgesundheitsministerium.de/elektronische-patientenakte.html

[8] E-Health und mHealth.
Last Accessed: 02.08.2022. [Online]. Available: https://www.mwv-berlin.de/meldung/!/id/45

[9] (2020) Gesetz zum Schutz elektronischer Patientendaten in der Telematikinfrastruktur (Patientendaten-Schutz-Gesetz – PDSG).
Last Accessed: 22.07.2022. [Online]. Available: https://www.bundesgesundheitsministerium.de/fileadmin/Dateien/3_Downloads/Gesetze_und_Verordnungen/GuV/P/PDSG_bgbl.pdf

[10] N. Pohlmann, *Cyber-Sicherheit: Das Lehrbuch für Konzepte, Prinzipien, Mechanismen, Architekturen und Eigenschaften von Cyber-Sicherheitssystemen in der Digitalisierung*, 2nd ed. Wiesbaden: Springer-Verlag, 2022, iSBN: 978-3-658-36243-0.

[11] J. Schwenk, *Sicherheit und Kryptographie im Internet: Theorie und Praxis*, 5th ed. Wiesbaden: Springer-Verlag, 2020, iSBN: 978-3-658-29260-7.

[12] P. D. C. Eckert, *IT-Sicherheit Konzepte - Verfahren – Protokolle*, 10th ed. Berlin: Bibliografische Information der Deutschen Nationalbibliothek, 2018, iSBN: 978-3-11-056390-0.

[13] (2022) BSI TR-03161 Anforderungen an Anwendungen im Gesundheitswesen.
Last Accessed: 11.07.2022. [Online]. Available: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03161/tr-03161.html

[14] (2021) Digitalisierung im Gesundheitswesen voranbringen (Patientendaten-Schutz-Gesetz – PDSG).
Last Accessed: 22.07.2022. [Online]. Available: https://www.bundesregierung.de/breg-de/suche/patientendaten-schutz-gesetz-1738402

[15] M. A. Pfannstiel, P. Da-Cruz, and H. M. Dekan, *Digitale Transformation von Dienstleistungen im Gesundheitswesen I Impulse für die Versorgung*. Springer Gabler, 2017.

[16] (2020) eHealth Monitor 2020.
Last Accessed: 24.07.2022. [Online]. Available: https://www.mckinsey.de/~/media/mckinsey/locations/europe%20and%20middle%20east/deutschland/news/presse/2020/2020-11-12%20ehealth%20monitor/ehealth%20monitor%202020.pdf

[17] (2022) BSI-Schrift 7164: Liste der zugelassenen IT-Sicherheitsprodukte und -systeme.
Last Accessed: 24.07.2022. [Online]. Available: https://www.bsi.bund.de/DE/Themen/Oeffentliche-Verwaltung/Zulassung/Liste-zugelassener-Produkte/liste-zugelassener-produkte_node.html

[18] (2021) IT-Sicherheit auf dem digitalen Verbrauchermarkt: Fokus Gesundheits-Apps.
Last Accessed: 24.07.2022. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/DVS-Berichte/gesundheitsapps.pdf

[19] Was sind personenbezogene Daten?
Last Accessed: 21.09.2022. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_de

[20] J. P. Christof Paar, *Kryptografie verständlich - Ein Lehrbuch für Studierende und Anwender*. Berlin: Springer-Verlag, 2016, iSBN: 978-3-662-49297-03.

[21] D. Wätjen, *Kryptographie Grundlagen, Algorithmen, Protokolle*, 3rd ed. Wiesbaden: Springer, 2018, iSBN: 978-3-658-22474-5.

[22] D. H. B. N. Prof. Dr. Albrecht Beutelspacher, Dr. Thomas Schwarzpau, *Kryptographie Grundlagen, Algorithmen, Protokolle*, 2nd ed. Wiesbaden: GWV Fachverlage GmbH, 2010, iSBN: 978-3-8348-0977-3.

[23] (2009) Standards for Efficient Cryptography 1 (SEC 1).
Last Accessed: 31.10.2022. [Online]. Available: https://www.secg.org/sec1-v2.pdf

[24] Message Authentication Codes.
Last Accessed: 11.10.2022. [Online]. Available: https://csrc.nist.gov/projects/message-authentication-codes

[25] Digital Signatures.
Last Accessed: 11.10.2022. [Online]. Available: https://csrc.nist.gov/Projects/digital-signatures

[26] (2006) The Secure Shell (SSH) Authentication Protocol.
Last Accessed: 25.10.2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4252

[27] (2001) SSH, The Secure Shell: The Definitive Guide.
Last Accessed: 29.10.2022. [Online]. Available: https://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch03_04.htm#ch03-83508.html

[28] What is SSH Public Key Authentication?
Last Accessed: 25.10.2022. [Online]. Available: https://www.ssh.com/academy/ssh/public-key-authentication

[29] M. Küller. (2019) FAQ Medizinprodukte.
Last Accessed: 05.08.2022. [Online]. Available: https://www.ig-nb.de/?tx_epxelo_file[id]=745338&cHash=315b475d8bf4b112e6a1da609616894b

[30] (2020) TÜV SÜD: Der Zweck macht Software zum Medizinprodukt.
Last Accessed: 05.08.2022. [Online]. Available: https://www.tuvsud.com/de-de/presse-und-medien/2020/november/der-zweck-macht-software-zum-medizinprodukt

[31] VERORDNUNG (EU) 2017/745 DES EUROPÄISCHEN PARLAMENTS UND DES RATES.
Last Accessed: 21.09.2022. [Online]. Available: https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX%3A32017R0745

[32] (2022) Das DiGA-Verzeichnis.
Last Accessed: 02.08.2022. [Online]. Available: https://diga.bfarm.de/de

[33] (2022) BSI TR-02102 Kryptographische Verfahren: Empfehlungen und Schlüssellängen.
Last Accessed: 11.07.2022. [Online]. Available: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html

[34] (2020) BSI TR-02103 X.509-Zertifikate und Zertifizierungspfadvalidierung.
Last Accessed: 16.07.2022. [Online]. Available: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02103/tr-02103.html

[35] B. Rancea. (2021) The Dangers of Third Party Code Dependency.
Last Accessed: 10.08.2022. [Online]. Available: https://ecommerce-platforms.com/articles/the-dangers-of-third-party-code-dependency

[36] V. Wegner. (2019) 4 Risks to consider when implementing third-party code.
Last Accessed: 10.08.2022. [Online]. Available: https://about.gitlab.com/blog/2019/07/16/third-party-code-risks/

[37] (2022) System.Security.Cryptography Namespace.
Last Accessed: 10.08.2022. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-6.0

[38] (2022) Overview of encryption, digital signatures, and hash algorithms in .NET.
Last Accessed: 11.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/security/cryptographic-services?redirectedfrom=MSDN

[39] Initialization Vector (IV).
Last Accessed: 11.10.2022. [Online]. Available: https://csrc.nist.gov/glossary/term/initialization_vector

[40] (2022) Announcement of Proposal to Revise Special Publication 800-38A .
Last Accessed: 11.10.2022. [Online]. Available: https://csrc.nist.gov/news/2022/proposal-to-revise-sp-800-38a

[41] (2006) CWE-329: Generation of Predictable IV with CBC Mode.
Last Accessed: 11.10.2022. [Online]. Available: https://cwe.mitre.org/data/definitions/329.html

[42] R. Fujita, T. Isobe, and K. Minematsu, "Ace in chains : How risky is cbc encryption of binary executable files ?" Cryptology ePrint Archive, Paper 2020/1159, 2020, https://eprint.iacr.org/2020/1159. [Online]. Available: https://eprint.iacr.org/2020/1159

[43] D. A. Werner, *Elliptische Kurven in der Kryptographie*. Berlin: Springer-Verlag, 2002, iSBN: 978-3-642-56351-5.

[44] (2008) An Interface and Algorithms for Authenticated Encryption.
Last Accessed: 29.10.2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5116

[45] (2003) A Conventional Authenticated-Encryption Mode.
Last Accessed: 19.10.2022. [Online]. Available: https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/proposed-modes/eax/eax-spec.pdf

[46] (2007) Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality.
Last Accessed: 19.10.2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

[47] (2003) Counter with CBC-MAC (CCM).
Last Accessed: 19.10.2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3610

[48] (2007) Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.
Last Accessed: 19.10.2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

[49] (2008) AES Galois Counter Mode (GCM) Cipher Suites for TLS.
Last Accessed: 19.10.2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5288

[50] (2008) Basic comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS).
Last Accessed: 19.10.2022. [Online]. Available: https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf

[51] RSA signature and encryption schemes: RSA-PSS and RSA-OAEP.
Last Accessed: 12.10.2022. [Online]. Available: https://www.cryptosys.net/pki/manpki/pki_rsaschemes.html

[52] (2003) Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1.
Last Accessed: 12.10.2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3447

[53] (2018) On the Security of the PKCS#1 v1.5 Signature Scheme.
Last Accessed: 11.10.2022. [Online]. Available: https://eprint.iacr.org/2018/855.pdf

[54] (2009) Comparison Research on Digital Signature Algorithms in Mobile Web Services.
Last Accessed: 20.10.2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5301198

[55] (2009) The Debian PGP disaster that almost was.
Last Accessed: 20.10.2022. [Online]. Available: https://rdist.root.org/2009/05/17/the-debian-pgp-disaster-that-almost-was/

[56] (2010) Non-weak DSA key? oxymoronic, perhaps.
Last Accessed: 20.10.2022. [Online]. Available: https://meyering.net/nuke-your-DSA-keys/

[57] (2016) Make Sure DSA Signing Exponentiations Really are Constant-Time.
Last Accessed: 20.10.2022. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2976749.2978420

[58] (2020) Big Numbers - Big Troubles: Systematically Analyzing Nonce Leakage in (EC)DSA Implementations.
Last Accessed: 20.10.2022. [Online]. Available: https://www.usenix.org/system/files/sec20-weiser.pdf

[59] (2010) Recommendation for Password-Based Key Derivation.
Last Accessed: 13.10.2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

[60] PBKDF2.
Last Accessed: 13.10.2022. [Online]. Available: https://cryptobook.nakov.com/mac-and-key-derivation/pbkdf2

[61] (2015) Password Hashing Competition.
Last Accessed: 13.10.2022. [Online]. Available: https://www.password-hashing.net/

[62] Argon2.
Last Accessed: 13.10.2022. [Online]. Available: https://cryptobook.nakov.com/mac-and-key-derivation/argon2

[63] (2015) Argon2: the memory-hard function for password hashing and other applications.
Last Accessed: 13.10.2022. [Online]. Available: https://www.password-hashing.net/argon2-specs.pdf

[64] (2015) High Parallel Complexity Graphs and Memory-Hard Functions.
Last Accessed: 13.10.2022. [Online]. Available: https://www.password-hashing.net/

[65] Rfc2898DeriveBytes Klasse.
Last Accessed: 13.10.2022. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/api/system.security.cryptography.rfc2898derivebytes?view=net-6.0

[66] (2011) A proposal for: Functionality classes for random number generators.
Last Accessed: 21.10.2022. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf

[67] (2022) FIPS 140-2 Validation.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/threat-protection/fips-140-validation

[68] (2022) FIPS 140-2-Validierung.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/de-de/windows/security/threat-protection/fips-140-validation

[69] (2001) Security Requirements for Cryptographic Modules.
Last Accessed: 21.10.2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

[70] RandomNumberGenerator.Create Method.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.randomnumbergenerator.create?view=net-6.0

[71] (2021) .NET cryptography model.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/security/cryptography-model

[72] (2015) Windows and Linux Random Number Generation Process: A Comparative Analysis.
Last Accessed: 21.10.2022. [Online]. Available: https://research.ijcaonline.org/volume113/number8/pxc3901710.pdf

[73] AesCryptoServiceProvider Class.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.aescryptoserviceprovider?view=net-6.0

[74] (2022) Overview of encryption, digital signatures, and hash algorithms in .NET.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/security/cryptographic-services

[75] (2022) Übersicht über Verschlüsselung, digitale Signaturen und Hashalgorithmen in .NET.
Last Accessed: 21.10.2022. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/standard/security/cryptographic-services

[76] DSA Class.
Last Accessed: 24.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.dsa?view=net-6.0

[77] ECDsa Class.
Last Accessed: 24.10.2022. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/api/system.security.cryptography.ecdsa?view=net-6.0

[78] D. Davis. (2001) Defective Sign and Encrypt in SMIME, PKCS#7, MOSS, PEM, PGP, and XML.
Last Accessed: 24.07.2022. [Online]. Available: https://theworld.com/~dtd/sign_encrypt/sign_encrypt7.html

[79] (2022) C-Sharp Garbage Collection.
Last Accessed: 04.10.2022. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/standard/garbage-collection/fundamentals?source=recommendations

[80] (2013) fixed Statement (C-Sharp Reference).
Last Accessed: 04.10.2022. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/f58wzh21(v=vs.100)?redirectedfrom=MSDN

[81] (2018) The Transport Layer Security (TLS) Protocol Version 1.3.
Last Accessed: 26.10.2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446

[82] (2020) Entwurf eines Gesetzes zum Schutz elektronischer Patientendaten in der Telematikinfrastruktur.
Last Accessed: 22.07.2022. [Online]. Available: https://www.bundesgesundheitsministerium.de/fileadmin/Dateien/3_Downloads/Gesetze_und_Verordnungen/GuV/P/Referentenentwurf_Patientendaten-Schutzgesetz__PDSG.pdf

[83] (2022) Technische Richtlinie BSI TR-03116 Kryptographische Vorgaben für Projekte der Bundesregierung.
Last Accessed: 26.10.2022. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03116/BSI-TR-03116-4.pdf?__blob=publicationFile&v=1

[84] H. Krawczyk. (2001) The Order of Encryption and Authentication for Protecting Communications.
Last Accessed: 24.07.2022. [Online]. Available: https://www.iacr.org/archive/crypto2001/21390309.pdf