# OpenDear

Open-source
Double Encryption And Re-encryption

# Programm zum quelloffenen Verschlüsseln, Umschlüsseln und Signieren

Benutzerhandbuch

- OpenPGP-Verschlüsselung mit RSA-Schlüssellängen von 2048 bis 16384 Bit.
- Optional auf OpenPGP-SmartCards gespeicherte, nicht auslesbare Schlüssel.
- Optional post-quanten-sichere symmetrische Verschlüsselung mit AES-256.

Diese Software soll Bestandteil einer dezentralen und quelloffenen Alternative zur Telematikinfrastruktur im deutschen Gesundheitswesen sein, insbesondere zur Kommunikation im Medizinwesen (KIM) und zur elektronischen Patientenakte (ePA). Noch ist sie unvollständig. Sie wird einsatzbereit sein, wenn die Programmversion mit der Ziffer 1 oder höher beginnt.

Das ist das Handbuch zu Programmversion 0.9.3 vom 06.09.2020.

Programmaktualisierungen sind unter der Adresse https://github.com/dasSubjekt/OpenDear ca. einmal wöchentlich geplant.

Kontakt: praxis@matthias-gloeckner.de
https://twitter.com/dasSubjekt

# Inhaltsverzeichnis

# 1. Philosophie und Einleitung

»Einen Brunnen graben just an der Stelle, wo man gerade steht.«

Mit diesem Satz bezieht der Schriftsteller Theodor Fontane (1819 bis 1898) sich in seinem Roman »Der Stechlin« auf die zu jener Zeit populäre »Wasserkur« des Pfarrers Sebastian Kneipp. Seine Romanfigur des Pastor Lorenzen lässt er anfügen: »Nicht so ganz unbedingt mit dem Neuen. Lieber mit dem Alten, soweit es irgend geht, und mit dem Neuen nur, soweit es muss.« Die Kneipp-Kur steht heute im bundesweiten Verzeichnis des immateriellen Kulturerbes. Populär sind dagegen andere, vor allem neuere, Heilmethoden. Also doch Altes zum Kulturerbe und dann lieber mit dem Neuen?

Das »soweit es muss« spricht Fontane auch mit fragendem Blick auf die zu jener Zeit sich entwickelnde Sozialdemokratie aus. Heute ist »soweit es muss« als ein Standpunkt in Diskussionen über die Digitalisierung zu hören. Aus »Wasser auf die Mühlen der Sozialdemokratie« am Ende des 19. Jahrhunderts scheint »Wasser auf die Mühlen der Digitalisierung« am Beginn des 21. Jahrhunderts geworden zu sein.

Wasser-Metaphern funktionieren noch immer. So haben symmetrische kryptographische Schlüssel weniger Bits als asymmetrische kryptographische Schlüssel. Der Grund dafür ist, dass erstere dicht gepackt sind wie Wasser bei 4 °C, letztere sind der Pulverschnee[1] der Primzahlen in der Luft aller teilbaren Zahlen. Bit kann wie Liter entweder die Maßeinheit für eine konkrete Menge sein oder das »Hohlmaß« für die maximal mögliche Menge.

Mit meiner Software »OpenDear« will ich der Diskussion um die Digitalisierung im deutschen Gesundheitswesen und speziell um die Telematikinfrastruktur[2] ein Maß an neuen Informationen hinzufügen. Falls es stimmt, dass Patientin und Patient Eigentümer ihrer Daten bleiben sollen, weshalb ist dann die Speicherung kryptographischer Schlüssel und der mit ihnen kodierten Daten in einer zentralisierten Telematikinfrastruktur besser als die dezentrale Lösung, die ich hier vorstelle?

Bereits im Jahr 2007 schlug der Hausarzt Wilfried Deiß im Deutschen Ärzteblatt[3] eine Punkt-zu-Punkt-Kommunikation aller Beteiligten des Gesundheitswesens über ein sicheres Netz vor. Mit meinem Projekt biete ich eine konkrete, praxistaugliche und, wie ebenfalls von ihm gefordert, »schlanke Lösung« zu dieser Idee an.

Zugleich möchte ich angesichts intensiver Forschung an Quantencomputern die Diskussion darüber fördern, über welchen Zeitraum hinweg ein »sicheres Netz« noch als sicher gelten kann. Vor 122 Jahren wurde »Der Stechlin« geschrieben. Vermutlich wesentlich früher als in 122 Jahren werden Quantencomputer alles, was wir mit den heutigen asymmetrischen bzw.

1 https://de.wikipedia.org/wiki/Ulam-Spirale

2 https://de.wikipedia.org/wiki/Telematikinfrastruktur

3 https://www.aerzteblatt.de/archiv/57836/Projekt-Elektronische-Gesundheitskarte-Fuchs-statt-Monster

hybriden Verfahren verschlüsseln und durch das Internet schicken, auch für Unbefugte lesbar machen können.[4]

Dabei darf Datensicherheit nicht vom Besitz der größeren technischen oder intellektuellen Fähigkeiten abhängen, nach dem Motto: wer das Rätsel löst, dem gehören die Daten. Datensicherheit soll auf dem Besitz eines Geheimnisses beruhen, das auch mit großen intellektuellen und technischen Möglichkeiten praktisch nicht erraten werden kann.

So halte ich es  für unabdingbar, dass Patientin und Patient die kryptographischen Schlüssel zum sicheren Speichern ihrer Daten selbst erzeugen und selbst verwalten können. Höchstens sogenannte öffentliche asymmetrische kryptographische Schlüssel gehören in die Hände einer zentralen Stelle, die ihre Veröffentlichung erleichtert und die Zugehörigkeit der Schlüssel zu einer bestimmten Person zertifiziert.

Wer dann immer noch die Hoheit über seine Gesundheitsdaten in die gesetzliche Betreuung der Telematikinfrastruktur übergeben wollte, könnte es durch freiwilliges Übermitteln seiner selbst erzeugten privaten kryptographischen Schlüssel an deren Betreiber tun.

Anders als die Telematikinfrastruktur ist meine Software quelloffen, so dass alle Interessierten ihre Funktionen nachvollziehen, auf Fehler hinweisen, Verbesserungen vorschlagen und eigene Versionen davon ableiten können.

Was die Software nicht kann, ist

1. Auf den ver- und entschlüsselnden Computern Daten und kryptographische Schlüssel gegen Viren, Trojaner und Hintertüren des Betriebssystems schützen,

2. Garantien oder Haftungsansprüche gegen mich begründen und

3. selbst Viren, Trojaner oder Hintertüren enthalten.

Verschlüsselung bedeutet nicht, dass die vertraulichen Schlüssel nicht aus dem Computer oder aus einem unsicheren Aufbewahrungsort entwendet werden können. Das muss nicht durch Trojaner geschehen. Auch von Windows weiß niemand, was es tut oder in Zukunft tun wird.

Der einzige Weg, wie kryptographische Schlüssel mit sehr hoher Wahrscheinlichkeit vor Diebstahl (und zugleich vor Wiederherstellung nach Verlust) geschützt sind besteht darin, sie auf einem Hardware-Token, z.B. einer OpenPGP-SmartCard erzeugen zu lassen. Ein Hardware-Token führt insbesondere die Algorithmen der Signatur und der Entschlüsselung selbst aus, so dass die dazu verwendeten privaten Schlüssel nicht ausgelesen werden müssen. Zudem verhindert die Hardware einer OpenPGP-SmartCard, dass private Schlüssel überhaupt ausgelesen werden können. OpenDear wird OpenPGP-SmartCards unterstützen.

---

4 https://www.aerzteblatt.de/nachrichten/99034/Langfristige-sichere-Speicherung-von-Gesundheitsdaten-laut-IT-Experten-im-Augenblick-nicht-gewaehrleistet

OpenDear enthält die Ver- und Entschlüsselungsalgorithmen (bisher) nicht selbst, sondern nutzt die Kryptographie-Schnittstelle »Next Generation« (CNG) von Windows. Das ist eine Weiterentwicklung derjenigen Schnittstelle, die vor 20 Jahren im Verdacht stand, einen Zweitschlüssel der National Security Agency der Vereinigten Staaten (NSA) zu enthalten[5]. Ob der NSA-Schlüssel mit weiterentwickelt wurde, ist nicht bekannt.

Mein Programm selbst kann keine Viren enthalten, da ich nur die Textdateien mit dem Quellkode zur Verfügung stelle. Wie daraus ein ausführbares Programm wird, steht in Kapitel 2 dieser Anleitung.

Beim Schreiben der Software und dieser Anleitung ist mir manches deutlicher geworden. Die wichtigsten Punkte sind:

1. Die Digitalisierung verlangt von Praxisinhaber*innen ein großes Wissen über Informationstechnik und Datenschutz, das sie sich teils selbst aneignen, teils von externen Berater*innen einkaufen müssen. Unter dieser Voraussetzung kann ihnen auch die Bedienung eines Programms wie des vorliegenden zugetraut werden.

2. Digitales Grundwissen gehört heute zur Allgemeinbildung. Wer irgendeine Art gesellschaftlicher Verantwortung trägt, sollte mit Kerckhoffs' Prinzip[6] und mit den Unterschieden zwischen symmetrischer, asymmetrischer und hybrider Verschlüsselung vertraut sein.

3. Asymmetrische Verschlüsselung ist schon jetzt leichter zu brechen als symmetrische Verschlüsselung. Mit fortschreitender Entwicklung von Quantencomputern wird sich dieser Trend beschleunigen. Das macht es immer weniger angemessen, beide Verfahren in demselben Begriff von »Datensicherheit« zusammenzufassen.

4. Wer Gesundheitsdaten keinesfalls in die Hände US-amerikanischer Konzerne geraten lassen will, sollte ein quelloffenes und freies Betriebssystem wie Linux nutzen.

Man darf meiner Software ansehen, dass ich vor der Psychologie zwei Jahre lang Informatik studiert habe und dass ich ein Befürworter der Digitalisierung nur insoweit bin, wie sie auf naturwissenschaftlicher Bildung und Einsicht beruht.

Im Verlauf von »Der Stechlin« verändert sich gemäß dessen langsamer, aber fließender Natur die Einstellung von Pastor Lorenzen gegenüber Alt und Neu. In dem zentralen Dialog bei zwei Dritteln des Buchs sagt er:

»Ich habe mich dagegen gewehrt, als das Eis aufgeschlagen werden sollte, denn alles Eingreifen oder auch nur Einblicken in das, was sich verbirgt, erschreckt mich. Ich respektiere das Gegebene. Daneben aber freilich auch das Werdende, denn eben dieses Werdende wird über kurz oder lang abermals ein Gegebenes sein. Alles Alte, soweit es Anspruch darauf hat, sollen wir lieben, aber für das Neue sollen wir recht eigentlich leben.«

---

5 https://de.wikipedia.org/wiki/NSAKEY

6 https://de.wikipedia.org/wiki/Kerckhoffs%E2%80%99_Prinzip

# 2. Das Programm zum Laufen bringen

## 2.1. Warum so kompliziert?

Als Benutzer des Betriebssystems Windows sind wir es gewohnt, eine Installationsdatei herunter zu laden, sie mit Doppelklick zu starten und die Nachfrage, ob wir wirklich wissen, was wir tun, mit dem gewohnten Klick auf »ja« zu beantworten. Für das Programm OpenDear gelten andere Regeln.

Wollte man wirklich nachvollziehen, was so eine Installationsdatei und das von ihr installierte Programm auf unserem Computer tun, müsste man großen technischen Aufwand betreiben. Funktioniert das Programm tatsächlich so, wie seine Dokumentation es beschreibt? Schickt es Daten in das Internet, ohne mich darüber zu informieren? Enthält es einen Computervirus, den der Virenscanner nicht erkennt? Für diese Zwecke gibt es spezialisierte Computerprogramme, sogenannte Disassembler und Decompiler. Jedoch braucht man zu ihrer Benutzung eine fortgeschrittene Programmiererausbildung. Außerdem hat die Anwendung solcher Programme auf kommerzielle Software rechtliche Hürden.

Um Transparenz und Sicherheit den Vorrang zu geben, habe ich für mein Programm eine Entscheidung getroffen, die es wesentlich einfacher macht, die eben genannten Fragen zu beantworten. Der Preis dafür ist ein etwas komplizierter und langwieriger Weg, das Programm zum ersten Mal auf dem eigenen Computer zum Laufen zu bringen. Als willkommenen Nebeneffekt dieser Umstände können auch Benutzer*innen ohne Programmierkenntnisse einige Programmfunktionen selbst anpassen. Beispielsweise, ob die Benutzeroberfläche auf Deutsch oder Englisch beschriftet ist.

Aus den Begriffen Disassembler und Decompiler lässt sich erahnen, dass es sich um die Gegenteile von Assemblern und Compilern handelt. Assembler sind sehr nah an der Maschinensprache des Computerprozessors angesiedelt und wandeln eine so genannte Assemblersprache in Prozessorbefehle um. Mit jedem Jahrzehnt der Computer- und Softwareentwicklung grenzte sich ihre Verwendung immer weiter auf Spezialbereiche ein und sie wurden durch Compiler für höhere Programmiersprachen ersetzt. Wer OpenDear nutzen möchte, braucht also einen Compiler, und zwar einen für die Programmiersprache C# (gesprochen wie »see sharp« - sieh scharf). Anstelle einer Installationsdatei muss der Quelltext des Programms aus dem Internet heruntergeladen werden. Der Compiler übersetzt diesen dann in  Prozessorbefehle.

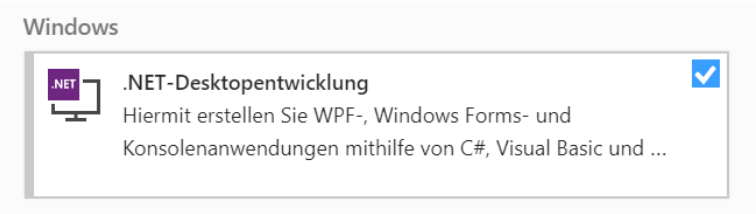
Bis vor einigen Jahren war das ungefähr so kompliziert, wie es klingt. Dank Microsoft (das muss hier einmal lobend gesagt werden) ist es einfacher geworden.

## 2.2. Herunterladen von Visual Studio Community

Um den Quellkode auf übersichtliche Weise anzuzeigen, bei Bedarf daran Änderungen vorzunehmen und schließlich ein lauffähiges Programm daraus zu erzeugen, wird eine sogenannte Integrierte Entwicklungsumgebung benötigt. Die Abkürzung dafür ist »IDE«, von englisch »Integrated Development Environment«.

In diesem Fall ist das »Visual Studio Community 2019«. Diese Entwicklungsumgebung kann von der Adresse https://visualstudio.microsoft.com/de/vs/ heruntergeladen werden. Bitte achten Sie darauf, nicht versehentlich die kostenpflichtige Professional- oder Enterprise-Version herunterzuladen. Da es sich tatsächlich um eine »Umgebung« handelt, nicht nur um ein Programm, kann die Installation bis zu einer Stunde oder länger dauern. Manchmal kommt es dabei zu kleineren Fehlern, die ignoriert werden können.

Nicht alle der zahlreichen Bestandteile müssen installiert werden. Benötigt wird die ».NET-Desktopentwicklung«, bei der ein Häkchen zu setzen ist:




Wer Visual Studio Community länger als 30 Tage nutzen möchte, muss sich mit einer E-Mail-Adresse bei Microsoft registrieren. Microsoft hält sich bedeckt, welche Nutzerdaten es aus Visual Studio ausliest und in den Vereinigten Daten oder an dritten Orten speichert. Deshalb empfiehlt es sich, zeitnah die ausführbare Programmdatei zu erzeugen.

Wenn das Programm nicht nur ausprobiert wird, sondern mit vertraulichen Daten gearbeitet werden soll, ist es sicherer, die ausführbare Datei »OpenDear.exe« immer direkt zu starten, nicht aus Visual Studio heraus.

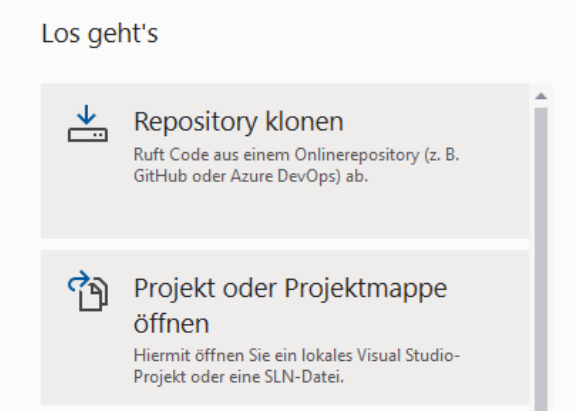
### 2.3. Herunterladen des Quellkode

Linus Torvalds ist der Ideen- und Namensgeber des Betriebssystems Linux. Er ist jedoch auch Ideen- und Namensgeber einer Versionsverwaltung für Software namens Git. Zumindest scherzte er, dass Git (Englisch für Depp) nach ihm benannt sei.[7]

Ein Unternehmen aus San Francisco entwickelte Git zu einem Speicherdienst für Softwareprojekte namens Github weiter. OpenDear steht dort zum Herunterladen bereit. Die Adresse ist

https://github.com/dasSubjekt/OpenDear .

Am einfachsten geht das Herunterladen, indem man auf der Startseite von Visual Studio Community »Repository klonen« wählt, die obige Adresse einträgt und auf »Klonen« klickt.




Alternativ können alle benötigten Dateien in Form eines Zip-Archivs von Github

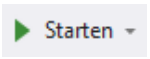
7 https://de.wikipedia.org/wiki/Git#Name

heruntergeladen werden. Anschließend lässt sich dieses mit der rechten Maustaste anklicken, auf der Festplatte entpacken und mit »Projekt oder Projektmappe öffnen« anzeigen.

Oder man sucht im entpackten Verzeichnis die Datei »OpenDear.sln« und öffnet sie per Doppelklick.

### 2.4. Ein lauffähiges Programm erzeugen

In Visual Studio Community startet man das Programm mit der Taste F5 oder mit einem Klick auf den grünen Pfeil.




Erläuterungen zum Unterschied zwischen Debug- und Releasekonfiguration und eine ausführliche Anleitung zum Umgang mit der erstellten ausführbaren Datei folgen zu einem späteren Zeitpunkt.

Unter Projekt => Build => Ausgabe => Ausgabepfad ist der relative Speicherort innerhalb des Quellkode-Verzeichnisses zu finden, an dem die ausführbare Datei »OpenDear.exe« erzeugt wurde. Innerhalb des Quellkode-Verzeichnis ist das in der Regel der Pfad »\OpenDear\OpenDear\bin\x64\Release«. Es genügt, diese ausführbare Datei und das Unterverzeichnis »Assets« mit seinem Inhalt an einen beliebigen Ort zu kopieren. Dort kann das Programm nun immer direkt gestartet werden, Visual Studio wird nicht mehr benötigt.

Zur Ausführung unter Windows benötigt OpenDear das NetFramework 4.7.2 oder höher. Falls es nicht schon auf dem Computer vorhanden war, wird es automatisch als ein Teil von Visual Studio Community installiert. Um OpenDear auf weiteren Windows-Computern ausführen zu können, muss zuvor gegebenenfalls das NetFramework 4.7.2 von der entsprechenden Microsoft-Seite aus der Spalte »Runtime«[8] heruntergeladen und auf dem Computer installiert werden. Fehlt es, erscheint beim Versuch des Programmstarts der entsprechende Hinweis.

Wer ein anderes Programmsymbol als das voreingestellte wünscht, kann eine Symboldatei z.B. von http://www.iconarchive.com/ herunterladen und in Visual Studio Community unter Projekt => OpenDear-Eigenschaften => Anwendung => Symbol festlegen. Die Festlegung für das kleine Symbol am Programmfenster oben links erfolgt separat in der Datei »MainWindow.xaml«.

Als Sprache der Benutzeroberfläche kann entweder Deutsch oder Englisch gewählt werden. Dazu muss unter Projekt => OpenDear-Eigenschaften => Build => Symbole für bedingte Kompilierung als Schlüsselwort entweder DEUTSCH oder ENGLISH eingetragen werden.

8 https://dotnet.microsoft.com/download/visual-studio-sdks

### 2.5. Quellkode individuell anpassen

Den Möglichkeiten der Anpassung des Quellkodes an eigene Bedürfnisse sind keine Grenzen gesetzt, solange das die kryptographischen Algorithmen nicht verändert. Vorschläge hierzu ergänze ich später.

### 2.6. Lizenzen

Die Lizenz für Visual Studio Community[9] räumt Nutzern weitreichende Rechte ein. Dafür »zahlen« sie möglicherweise mit nicht genau bezeichneten Nutzerdaten.

Der OpenPGP-Standard RFC 4880 ist seit Ende der 1990-er Jahre lizenzfrei[10]. Sowohl der Advanced Encryption Standard als auch das RSA-Kryptosystem sind lizenzfrei. Das Patent für den RSA-Algorithmus erlosch am 21. September 2000. Beim AES-Algorithmus war die Freiheit von patentrechtlichen Ansprüchen eine der Bedingungen des Ausschreibungsverfahrens.

Das OpenDear-Programmsymbol stammt aus der Sammlung »Free Oktoberfest Icons«[11] der Ergosign GmbH[12]. Die Symbole innerhalb des Programms sind den Sammlungen »My Secret Icons«[13] und »Lovely Website Icons«[14] der Designerin Tanya Suhodolska[15] entnommen. Zusätzlich findet das »Sailingship Icon«[16] des Künstlers cemagraphics Verwendung. Alle Autor*innen erlauben die nichtkommerzielle Nutzung ihrer Werke unter der Bedingung einer Verlinkung auf sie als Urheber.

Beim Programmieren der OpenPGP-Funktionen habe ich ausgiebig in der »Bouncy Castle C# Distribution«[17] nachgeschaut. Aus dem Projekt InfInt[18] habe ich Programmkode für die Darstellung kryptographischer Schlüssel in Dezimalschreibweise übernommen, grundlegende Funktionen des ViewModel aus dem Projekt mvvmlight[19]. Bouncy Castle und mvvmlight verwenden die MIT-Lizenz[20], die das Nutzen, Verändern, Weiterverbreiten und Verkaufen der so lizenzierten Software erlaubt, Garantie und Haftung jedoch ausschließt.

---

9 https://visualstudio.microsoft.com/de/license-terms/mlt031819/

10 https://www.openpgp.org/about/

11 http://www.iconarchive.com/show/free-oktoberfest-icons-by-ergosign.html

12 https://www.ergosign.de

13 https://iconarchive.com/show/my-secret-icons-by-artdesigner.html

14 https://iconarchive.com/show/lovely-website-icons-by-artdesigner.html

15 https://artdesigner.me/

16 https://www.deviantart.com/cemagraphics/art/Sailingship-Icon-96460904

17 https://github.com/bcgit/bc-csharp/

18 https://github.com/sercantutar/infint

19 https://github.com/lbugnion/mvvmlight

20 https://opensource.org/licenses/MIT

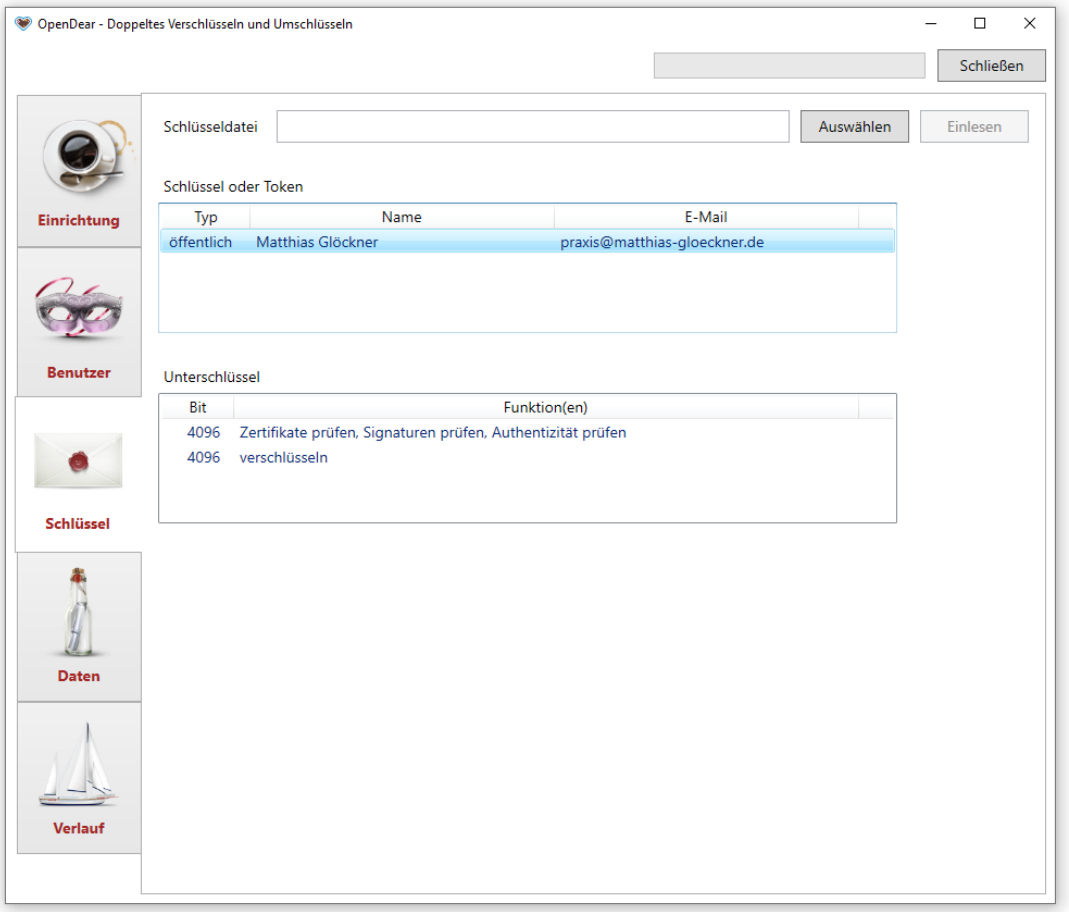
Eine eventuell später zu ergänzende Version von OpenDear, die unter dem Betriebssystem Linux lauffähig ist, würde die Software-Plattform .NET Core[21] und die Benutzeroberfläche AvaloniaUI[22] benötigen. Beide stehen ebenfalls unter der MIT-Lizenz[23].

Diesem Muster bin ich gefolgt und veröffentliche OpenDear unter der MIT-Lizenz.

# 3. Das Programm bedienen

## 3.1. Aufbau der Benutzeroberfläche

Am linken Rand des Programmfensters sind untereinander fünf Register angeordnet: »Einrichtung«, »Benutzer«, »Schlüssel«, »Daten« und »Verlauf«. Jedes dieser Register kann per Mausklick auf das entsprechende Symbol in den Vordergrund geholt werden.




Diese Registerkarten werde ich bis Ende September 2020 mit Funktionen füllen und sie dann in diesem Handbuch erläutern.

21 https://github.com/dotnet/core

22 https://github.com/AvaloniaUI/Avalonia

23 https://de.wikipedia.org/wiki/MIT-Lizenz

Bisher kann das Programm extern erzeugte OpenPGP-Schlüssel einlesen, anzeigen und private Schlüssel nach Eingabe ihrer Passphrase entschlüsseln. Weiterhin kann es Hardwaretoken wie den Nitrokey auslesen und die darauf befindlichen öffentlichen Schlüssel anzeigen.

Eine Schlüsseldatei für die Registerkarte »Schlüssel« oder eine Eingabedatei für die Registerkarte »Daten« kann über die jeweilige Schaltfläche »Auswählen« auf einem Datenträger gesucht werden. Es ist auch möglich, die Datei mit Ziehen und Ablegen aus dem Windows-Explorer in das Textfeld für den Dateipfad zu übernehmen. Sollte beim Ziehen das jeweilige Register nicht im Vordergrund sein, kann es aktiviert werden, indem man den Mauszeiger darüber bewegt.

### 3.2. Datenbank anlegen

### 3.3. Benutzer verwalten

### 3.4. Asymmetrische Schlüssel verwalten

Asymmetrische Schlüssel können verschiedenen Zwecken dienen. Deshalb werden meist zwei oder mehr von ihnen zusammengefasst. Eine solche Zusammenfassung mehrerer Unterschlüssel desselben Benutzers bezeichnet man als Sicherheits-Token, Schlüsselring oder Schlüssel(-Datei). Ein Blick in Listen möglicher Schlüsselarten bei der Internet Engineering Task Force[24] oder der Wikipedia[25] zeigt die wichtigsten Einsatzzwecke asymmetrischer Schlüssel:

- Verschlüsseln[26] ist das Umwandeln von Daten in nicht interpretierbare Zeichenfolgen mit dem Ziel, die Vertraulichkeit zu wahren. Der umgekehrte Vorgang ist das Entschlüsseln.

- Signieren[27] ist ein Vorgang, der garantiert, dass man das Dokument oder die Daten entweder selbst verfasst hat oder dass man aus anderen Gründen »unterschreibt«, z.B. um Kenntnisnahme oder Zustimmung zu zeigen (Authentizität). Zugleich werden die Daten dabei gegen nachträgliche Veränderung geschützt (Integrität). Der umgekehrte Vorgang ist das Überprüfen (Verifizieren) einer Signatur.

- Zertifizieren[28] bedeutet, einen öffentlichen Schlüssel einer anderen Person zu signieren, um damit einen Zertifizierungspfad[29] und somit transitives Vertrauen[30]

24 https://tools.ietf.org/html/rfc4880#section-5.2.3.21

25 https://en.wikipedia.org/wiki/Cryptographic_key_types#Key_types

26 https://de.wikipedia.org/wiki/Hybride_Verschl%C3%BCsselung

27 https://de.wikipedia.org/wiki/Digitale_Signatur

28 https://de.wikipedia.org/wiki/Public-Key-Zertifikat

29 https://en.wikipedia.org/wiki/Chain_of_trust

30 https://www.karteikarte.com/card/1541887/transitives-vertrauen-und-das-web-of-trust

herzustellen. Der umgekehrte Vorgang ist das Überprüfen (Verifizieren) eines Zertifikats. Ist kein Schlüsselpaar extra zum Zertifizieren vorgesehen, kann dafür das Signaturschlüsselpaar verwendet werden.

- Authentifizieren[31] bedeutet, eine Zugangsberechtigung zu erhalten, z.B. sich über das Internet auf einen Server einzuloggen. Ist kein Schlüsselpaar extra zum Authentifizieren vorgesehen, kann dafür das Signaturschlüsselpaar verwendet werden. Diese Schlüsselart wird von OpenDear aus Gründen der Kompatibilität mit verwaltet, aber nicht benötigt.

Verschiedene Schlüsselpaare (aus jeweils privatem und öffentlichem Schlüssel) für verschiedene Funktionen lassen es zu, abweichende Gültigkeitszeiträume zu definieren, z.B. die Authentifizierung häufiger zu wechseln als die Verschlüsselung und diese wiederum häufiger als Signatur und Zertifizierung. Auch lässt sich so einer der Schlüssel an Vertraute weitergeben (beispielsweise an einen Arbeitgeber, der den Zugriff auf verschlüsselte Arbeitsergebnisse braucht), ohne dass die anderen Funktionen mit kompromittiert werden. Man soll nie Daten mit demselben (privaten) Schlüssel signieren, mit dem man sie (per öffentlichem Schlüssel) verschlüsselt. Mathematische Wege, auf denen das sich für einen Angriff ausnutzen ließe, sind zwar kompliziert und von der konkreten Software abhängig. Aber sicher ist sicher.

OpenDear verwendet ausschließlich asymmetrische Schlüssel, die im OpenPGP-Standard und für das RSA-Verfahren erzeugt wurden. Die Schlüssel können mit anderer Software wie Gpg4win[32] oder dem Enigmail-Add-On für Thunderbird[33] erzeugt und dann importiert werden. Mit einer der nächsten Programmaktualisierungen wird OpenDear diese Schlüssel auch selbst erzeugen und exportieren können.

Mit einer der nächsten Aktualisierungen werden alle Bestandteile eines Schlüssels auf dem Bildschirm angezeigt werden können. Wer nicht in der Lage ist, sich die beiden Primzahlen p und q seiner RSA-Schlüssel anzeigen zu lassen, hat nicht die Kontrolle über die damit verschlüsselten Daten. Die einzige Ausnahme von dieser Regel sind nachweislich auf einem Hardware-Token erzeugte, nicht auslesbare Schlüssel.

Während symmetrische Schlüssel nur einen Wert besitzen, bestehen asymmetrische Schlüssel für das RSA-Verfahren aus mehreren Werten. Im Einzelnen sind das:

| RSA-Wert | Beschreibung | Geheimhaltung |
|---|---|---|
| p | eine Primzahl | geheim |
| q | eine andere Primzahl | geheim |

31 https://de.wikipedia.org/wiki/Authentifizierung

32 https://www.gpg4win.de/

33 https://addons.thunderbird.net/de/thunderbird/addon/enigmail/

| RSA-Wert | Beschreibung | Geheimhaltung |
|---|---|---|
| p * q | Das Produkt aus p und q. Diese Zahl wird auch mit dem Buchstaben N bezeichnet oder, mit Bezug auf die Division mit Rest, als Modul[34]. | öffentlich |
| e | Der öffentliche Exponent oder Verschlüsselungs-exponent. Er muss eine zu dem Wert von (p - 1) * (q - 1) teilerfremde Zahl sein, die weder so klein (z.B. 3) ist, dass das Verfahren angreifbar wird, noch so groß, dass dieser Exponent während der Berechnung zu unnötig riesigen Zahlen führt. Für e wird üblicherweise die Zahl 65537 verwendet. | öffentlich |
| d | Der Entschlüsselungsexponent ist das multiplikativ Inverse von e bezüglich des Moduls (p - 1) * (q - 1) und kann über den Euklid-Algorithmus für den größten gemeinsamen Teiler bestimmt werden. | geheim |
| dp | d mod (p – 1), ein Hilfswert. | geheim |
| dq | d mod (q – 1), ein Hilfswert. | geheim |
| invq | (1 / q) mod p, ein Hilfswert. | geheim |

Die Hilfswerte dp, dq, und invq beziehen sich auf ein Rechenverfahren mit dem Namen Chinesischer Restsatz[35]. Sie zu speichern und nicht jedes Mal neu zu berechnen steigert die Effizienz des Verfahrens.

Weitere Informationen zu asymmetrischen Schlüsseln stehen in Kapitel 4.7. »Eine kurze Geschichte der hybriden Verschlüsselung« und in Kapitel 4.8. »Das RSA-Kryptosystem nach Rivest, Shamir und Adleman«.

### 3.5. Symmetrische Schlüssel verwalten

### 3.6. Nachrichten ver- und entschlüsseln

Geplant ist die Ver- und Entschlüsselung sowie die Signatur und Verifizierung von OpenPGP-Nachrichten. Das soll einschließlich optionaler Dokumentenanhänge geschehen, sowohl auf lokalen Datenträgern als auch mit guter Kompatibilität zu OpenPGP-E-Mails.

Innerhalb des OpenPGP-Containers ist eine optionale zweite Verschlüsselung für Post-Quanten-Sicherheit geplant. Diese erfolgt per mit persönlich Bekannten offline vereinbartem und möglichst häufig erneuertem AES-256-Schlüssel. Zugleich wird hier durch Anhängen des öffentlichen Empfänger-Schlüssels die OpenPGP-Sicherheitslücke »heimliches Weiterleiten« geschlossen.

---

34 https://de.wikipedia.org/wiki/Division_mit_Rest#Modulo

35 https://de.wikipedia.org/wiki/RSA-Kryptosystem#RSA_mit_dem_Chinesischen_Restsatz

Die ausschließliche Kontrolle über die Daten behält der Schlüsselbesitzer, der im Sinn einer digitalen Patientenakte mittels Umschlüsseln am eigenen PC Daten für Dritte freigeben kann. Um unzulässige Änderungen der Daten dabei auszuschließen, werden alle ursprünglichen Signaturen des Dokuments übernommen.

### 3.7. Registerkarte »Verlauf«

Bisher wird hier die Programmversion angezeigt und ob OpenDear eine OpenSC-Bibiothek zur Kommunikation mit OpenPGP-SmartCards gefunden hat.

### 3.8. Geplante Programmfunktionen

In einer mittels des AES-256-Verfahrens verschlüsselten Datenbank auf dem lokalen Rechner werden die eigenen Schlüssel und die Kontaktdaten und Schlüssel von Kommunikationspartner*innen gespeichert. Der Zugang zur Datenbank kann entweder mit einem OpenPGP-Token wie dem NitroKey gesichert werden oder mit einem Passwort, aus welchem OpenDear den AES-Schlüssel mit der »Password-Based Key Derivation Function 2«[36] ableitet.

OpenDear wird zu privaten und öffentlichen OpenPGP-Schlüsseln kompatibel sein, die in externen Programmen wie Gpg4win oder dem Enigmail-Add-On für Thunderbird für das RSA-Verfahren erstellt wurden. AES-Schlüssel werden hingegen ein programmspezifisches Format haben.

Es wird möglich sein, RSA-Schlüsseldateien im OpenPGP-Format RFC 4880 einzulesen oder neu zu erstellen, sie zu verändern, zu signieren und für andere Programme zu exportieren. Analog können AES-256-Schlüssel erstellt, verändert, signiert und exportiert werden.

Nachrichten können im OpenPGP-Format verschlüsselt und signiert werden. Optional wird es innerhalb der OpenPGP-Verschlüsselung eine zweite Verschlüsselungsebene im Verfahren AES-256 geben, die zusätzliche Sicherheit bietet, aber nicht mehr mit OpenPGP kompatibel ist.

Zur besonders sicheren Aufbewahrung von RSA-Schlüsseln wird OpenDear mit OpenPGP-SmartCards kompatibel sein. Ich werde das Programm mit dem »Nitrokey Pro 2« und dem »Nitrokey Storage 2«[37] testen.

Angedacht ist, in OpenDear alle kryptographischen Algorithmen wie RSA, AES, SHA und die Erzeugung von Zufallszahlen quelloffen zu implementieren. Nutzer könnten dann zwischen quelloffener Verschlüsselung und Verschlüsselung durch das Betriebssystem wählen.

Es ist noch nicht entschieden, ob von OpenDear aus das Übertragen von RSA-Schlüsseln auf OpenPGP-SmartCards wie den Nitrokey und die Schlüsselerzeugung direkt in einer

36 https://de.wikipedia.org/wiki/PBKDF2

37 https://www.nitrokey.com/#comparison

OpenPGP-SmartCard möglich sein werden. Auf der Nitrokey-Homepage befinden sich jedoch die Anleitungen für die von meiner Software unabhängige sichere Schlüsselerzeugung[38] und den Schlüsselimport[39].

### 3.9. Nächster Aktualisierungstermin

Die nächste Programmaktualisierung ist für Samstag, den 12.09.2020 geplant.

### 3.10. Offene Fragen

- Da OpenDear zunächst nur für das Betriebssystem Windows programmiert wird, ist es nicht auf anderen Betriebssystemen oder auf Smartphones lauffähig.

- Die Zertifizierung der selbst erzeugten öffentlichen Schlüssel für eine Public-Key-Infrastuktur mit transitivem Vertrauen ist ungeklärt.

- Moderne Konzepte wie Elliptische-Kurven-Kryptographie, Perfect Forward Secrecy und Signal-Protokoll werden nicht unterstützt.

38 https://www.nitrokey.com/de/documentation/openpgp-create-on-device

39 https://www.nitrokey.com/de/documentation/openpgp-create-backup#key-import

# 4. Hintergrund: Kodieren und Verschlüsseln

## 4.1. Vertraulichkeit, Integrität, Verfügbarkeit

Informationssicherheit umfasst drei Schutzziele, die von einander weitgehend unabhängig sind: Vertraulichkeit, Integrität und Verfügbarkeit von Daten.

Vertraulichkeit wird durch Rechtsnormen wie § 203 StGB geschützt. Durch technische Mittel kann sie gefördert oder erzwungen werden. Verfügbarkeit ist selten gesetzlich geregelt, stützt sich jedoch gleichfalls auf technisch-organisatorische Maßnahmen wie das regelmäßige Anlegen von Sicherungskopien und das Vorhalten von Ersatz-Hardware.

Wesentlich weniger bekannt als die Tatsache, dass die Vertraulichkeit von Daten unter anderem mittels Verschlüsselung und ihre Verfügbarkeit unter anderem durch Redundanz[40] sichergestellt werden kann, ist die Bedeutung von Datenintegrität.

Datenintegrität ist unabhängig vom Inhalt, d.h. der Bedeutung von Daten und versucht zu verhindern, dass Daten unbemerkt beschädigt oder verändert werden. Die Beschädigung kann durch technische und menschliche Fehler oder mit Absicht geschehen. Datenintegrität hat mit den anderen beiden Schutzzielen gemeinsam, dass sie mit technischen Mitteln gefördert oder erzwungen werden kann. Die wichtigsten technischen Verfahren, um Datenintegrität sicherzustellen, sind Signaturen und Nachrichtenauthentifizierungskodes[41].

Auch sicher verschlüsselte Daten können beschädigt werden, weshalb Verschlüsselung immer mit Signatur kombiniert werden sollte. Auf der nächsten Seite folgt ein Beispiel für bewahrte Vertraulichkeit ohne Schutz der Datenintegrität. Einen Text aus Goethes »Faust«, zweiter Teil, habe ich mittels AES-256 verschlüsselt, nach dem Verschlüsseln die Daten in vier Blöcke geteilt und die Blöcke in der Reihenfolge 1–3–2–4 wieder zusammengefügt. Das Ergebnis war ohne technische Probleme zu entschlüsseln und ergab den folgenden Text. Die von mir rot hervorgehobenen Fehler haben jeweils die Länge eines AES-Blocks von 16 Byte. Erst die, hier fehlende, Signatur hätte zu einer Fehlermeldung geführt.

40 https://de.wikipedia.org/wiki/Redundanz_(Technik)

41 https://de.wikipedia.org/wiki/Message_Authentication_Code

Faust
Wohin der Weg? –

Mephistopheles
Kein Weg! Ins Unbetretene,
Nicht zu Betretende; ein Weg ans Unerbetene,
Nicht zu Erbittende. Bist du bereit? –
Nicht Schlösser sind, nicht Riegel wegzuschieben,
Von Einsamkeiten wirst umhergetrieben.
Hast du Begriff von öd' und Einsamkeit?

Faust
Du spartest, dächt' ich, solche Sprüche;
Hier wittert's nach der Hexenküche,
Nach einer längst vergangnen Zeit.
Mußt' ich nicht mit der Welt verkehren?
Das Leere lernen, Leeres lehren? –
Sprach ich vernünftig, wie ich's angeschaut,
Erklang der Widerspruch gedoppelt laut;
Mußt' ich sogar vor widerwärtigen Streichen
Zur Einsamkeit, zur Wildernis entweichen
Und, um nicht ganz versäumt, allein zu leben,
Mich doch zuletzt dem Teufel übergeben.

Mephistopheles
Und hättest du den Ozean durchschwommen,
Das Grenzenlose dort geschaut,
So sähst du dort doch Well' auf Welle kommen,
Selbst wenn es dir vorm Untergange **Ñ¼øp¢Ä#U£1[0ì“@R**tern.

Faust
Den Müttern! Trifft's mich immer wie ein Schlag!
Was ist das Wort, das ich nicht hören mag?

Mephistopheles
Bist du beschränkt, daß neues Wort dich stört?
Willst du nur hören, was du schon gehört?
Dich störe nichts, wie es auch weiter klinge,
Schon längst gewohnt der wunderbarsten Dinge.

Faust
Doch im Erstarren such' ich nicht mein Heil,
Das Schaudern ist der Menschheit bestes Teil;
Wie auch die Welt ihm das Gefühl verteure,
Ergriffen, fühlt er tief das Ungeheure.

Mephistopheles
Versinke denn! Ich könnt' auch sagen: steige!
's ist einerlei. Entfliehe dem Entstandnen
In der Gebilde losgebundne Reiche!
Ergetze dich am längst nicht mehr Vorhandnen;
Wie Wolkenzüge schlingt sich das Getreibe,
Den Schlüssel schwinge, halte sie vom Leibe!

Faust
Wohl! fest ihn fassend fühl' ich neue Stärke,
Die Brust erweitert, hin zum großen Werke.

Mephistopheles
Ein glühnder Dreifuß tut dir endlich kund,
Du seist im tiefsten, allertiefsten Grund.
Bei seinem Schein wi***‰6È6;°Ë]å#i'ûâþ** doch etwas. Sähst wohl in der Grüne
Gestillter Meere streichende Delphine;
Sähst Wolken ziehen, Sonne, Mond und Sterne –
Nichts wirst du sehn in ewig leerer Ferne,
Den Schritt nicht hören, den du tust,
Nichts Festes finden, wo du ruhst.

Faust
Du sprichst als erster aller Mystagogen,
Die treue Neophyten je betrogen;
Nur umgekehrt. Du sendest mich ins Leere,
Damit ich dort so Kunst als Kraft vermehre;
Behandelst mich, daß ich, wie jene Katze,
Dir die Kastanien aus den Gluten kratze.
Nur immer zu! wir wollen es ergründen,
In deinem Nichts hoff' ich das All zu finden.

Mephistopheles
Ich rühme dich, eh' du dich von mir trennst,
Und sehe wohl, daß du den Teufel kennst;
Hier diesen Schlüssel nimm. –

Faust
Das kleine Ding!

Mephistopheles
Erst faß ihn an und schätz ihn nicht gering.

Faust
Er wächst in meiner Hand! er leuchtet, blitzt!

Mephistopheles
Merkst du nun bald, was man an ihm besitzt?
Der Schlüssel wird die rechte Stelle wittern,
Folg ihm hinab, er führt**ÎP²Ži„ì…ó: ó9YÉ**r sehn,
Die einen sitzen, andre stehn und gehn,
Wie's eben kommt. Gestaltung, Umgestaltung,
Des ewigen Sinnes ewige Unterhaltung.
Umschwebt von Bildern aller Kreatur;
Sie sehn dich nicht, denn Schemen sehn sie nur.
Da faß ein Herz, denn die Gefahr ist groß,
Und gehe grad' auf jenen Dreifuß los,
Berühr ihn mit dem Schlüssel! –

Mephistopheles
So ist's recht!
Er schließt sich an, er folgt als treuer Knecht;
Gelassen steigst du, dich erhebt das Glück,
Und eh' sie's merken, bist mit ihm zurück.
Und hast du ihn einmal hierher gebracht,
So rufst du Held und Heldin aus der Nacht,
Der erste, der sich jener Tat erdreistet;
Sie ist getan, und du hast es geleistet.
Dann muß fortan, nach magischem Behandeln,
Der Weihrauchsnebel sich in Götter wandeln.

Faust
Und nun was jetzt? –

Mephistopheles
Dein Wesen strebe nieder;
Versinke stampfend, stampfend steigst du wieder.

Mephistopheles
Wenn ihm der Schlüssel nur zum besten frommt!
Neugierig bin ich, ob er wiederkommt.

### 4.2. Was ist ein kryptographischer Schlüssel?

Ein kryptographischer Schlüssel ist entweder eine Zahl oder eine Liste mehrerer Zahlen. Es sind immer positive ganze Zahlen, also ohne Minus und ohne Kommastellen. Diese positiven ganzen Zahlen müssen viel größer sein als alle Zahlen, die uns sonst im Alltag begegnen, um die Vertaulichkeit der mit ihnen verschlüsselten Daten zu garantieren. Der Advanced Encryption Standard beispielsweise verwendet als Schlüssel Zahlen zwischen 0 und 115.792.089.237.316.195.423.570.985.008.687.907.853.269.984.665.640.564.039.457.584.007.913.129.639.935.

### 4.3. Bits, Bytes und das Stellenwertsystem mit der Basis 16

Um zu wissen, woher die »krummen« Zahlen beim Umgang mit Computern kommen, ist es hilfreich, Zahlensysteme zu verstehen. Eines nicht mehr fernen Tages werden neuronale Netze und Quantencomputer flexibler mit Zahlen umgehen können. Aber vorerst kennen Computer nur die beiden Zustände »Strom fließt« und »Strom fließt nicht«. Jenseits der mittlerweile sprichwörtlichen 0 und 1 sehen Zahlen in einem Computer deshalb anders aus als die Zehner-, Hunderter- und Tausenderstellen in unserem Alltag.

Man könnte es bei der 0 und 1 des binären Zahlensystems belassen und anstelle 256-Bit-Verschlüsselung von 100000000-Bit-Verschlüsselung sprechen. Nur wäre das platzraubend und unübersichtlich. So hat es sich bewährt, dass das menschliche Gehirn und die 0 und 1 des Computers sich im Stellenwertsystem mit der Basis 16 treffen. Um nicht durcheinander zu geraten, stellt man solchen Hexadezimalzahlen eine 0x voran. Also 0x100-Bit-Verschlüsselung.

Ein wenig Mathematik: 256 = 10 * 10 * 2 + 10 * 5 + 6 = 16 * 16 * 1 + 16 * 0 + 0. Verwendet man Sechzehner-, Zweihundertsechsundfünfziger- und Viertausendsechsundneunzigerstellen anstelle der Zehner-, Hunderter- und Tausenderstellen, lässt sich daraus in kleinen »Happen« auf die interne Darstellung im Computer schließen und »krumme« Dezimalzahlen stehen als »schöne« Hexadezimal- oder Binärzahlen da.

Weil es für 10, 11, 12, 13, 14 und 15 keine einzelnen Ziffern gibt, verwendet man in der Hexadezimalschreibweise dafür die Buchstaben, A, B, C, D, E und F, wahlweise groß oder klein geschrieben. Ein unschönes Detail ist, dass »Byte« so klingt wie englisch »bite« = deutsch »Bissen« und auch davon abgeleitet ist. Allerdings hat es sich so entwickelt, dass man zwei Happen = Stellen in der hexadezimalen Schreibweise als ein Byte bezeichnet. Das Byte 10010110 im Computer entspricht also der Zahl 2 * 2 * 2 * 2 * 2 * 2 * 2 * 1 + 2 * 2 * 2 * 2 * 2 * 2 * 0 + 2 * 2 * 2 * 2 * 2 * 0 + 2 * 2 * 2 * 2 * 1 + 2 * 2 * 2 * 0 + 2 * 2 * 1 + 2 * 1 + 0 = 150 = 16 * 9 + 6 = hexadezimal 0x96. Eine Stelle hexadezimal fasst vier Nullen oder Einsen, d.h. vier Bits, der internen Computerschreibweise zusammen und kodiert so ein halbes Byte. Acht Bits gleich ein Byte. Ein Byte kodiert die gleiche Informationsmenge wie log(256) / log(10) ≈ 2,40824 Dezimalziffern. Und scherzhaft, deshalb wenig gebräuchlich, besteht ein Byte aus zwei Nibble. Ein Nibble ist gleich einer Hexadezimalziffer.

Zeit zur Auflockerung. Zur Erinnerung an den Science-Fiction-Schriftsteller Douglas Adams wird am 25. Mai jeden Jahres der »Handtuchtag«[42] begangen. In seinem Roman »Per Anhalter durch die Galaxis« wird ein Handtuch als »so ziemlich das Nützlichste« bezeichnet, was man auf Reisen durch die Galaxis mit sich führen kann. Die Antwort auf die Frage aller Fragen, die Frage »nach dem Leben, dem Universum und dem ganzen Rest« beantwortet der Roman mit »42«[43]. Zu Lebzeiten von Douglas Adams hatte der 25. Mai keine Verbindung zu ihm oder seinen Büchern. Erst nach Adams‘ Tod kam jemand auf die Idee, man könne doch hexadezimal rechnen 0x25 + 0x5 = 0x2a = 16 * 2 + 10 = 42.

Im Computer muss auch alles Nichtmathematische wie Texte, Bilder und Musik auf Nullen und Einsen zurückgeführt werden. So wird der Text »i love you« als 0x69 0x20 0x6c 0x6f 0x76 0x65 0x20 0x79 0x6f 0x75 gespeichert[44]. Für die Binärschreibweise dessen höre man das Lied »Analogpunk«[45] von Judith Holofernes.

### 4.4. Eine kurze Geschichte der symmetrischen Verschlüsselung

Schon der römische Feldherr Gaius Julius Caesar kannte Verschlüsselung. In den Nachrichten an seine Truppen, die nicht in die Hand des Feindes fallen durften, ersetzte er jeden Buchstaben durch den an 3. Stelle nachfolgenden Buchstaben. Anstelle »Telematikinfrastruktur« hätte Caesar »Whohpdwlnlqiudvwuxnwxu« geschrieben. Dieser sehr einfache und trotzdem einschüchternd wirkende Kode ist als Caesar-Verschlüsselung[46] bekannt.

Die im Verlauf der Jahrhunderte häufig unsichere, unsystematische und im Vergleich zu heute seltene Anwendung von Verschlüsselung erfuhr mit der Verbreitung der Telegraphie eine Systematisierung. Im Jahr 1883 veröffentlichte der niederländische Linguist Auguste Kerckhoffs (1835 bis 1903) seine Schrift »La cryptographie militaire« (deutsch »Die militärische Kryptographie«). Darin formulierte er einen Grundsatz für sichere Kommunikation, der bis heute gilt: Die Sicherheit des Verfahrens darf sich allein auf seine mathematische Stärke und auf die Geheimhaltung des Schlüssels stützen. Falls der Rest des Systems bekannt wird, insbesondere der Verschlüsselungsalgorithmus und die verwendete Hardware, darf kein Schaden entstehen. Im Gegenteil zu dem naiven Impuls, auch das Verschlüsselungsverfahren geheim zu halten, soll es veröffentlicht werden, so dass möglichst viele Experten es auf Schwächen untersuchen können.

Im Ersten Weltkrieg, 15 Jahre nach Kerckhoffs‘ Tod, trug die Entzifferung deutscher Funksprüche ab Frühjahr 1918 nach Ansicht einiger Historiker und Kryptologen dazu bei, dass es den Deutschen nicht gelang, Paris einzunehmen[47].

---

42 http://towelday.org/faq/#whydate

43 https://de.wikipedia.org/wiki/42_(Antwort)

44 https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

45 https://www.youtube.com/watch?v=LDbJ2Mx-I6o&t=112

46 https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung

47 https://de.wikipedia.org/wiki/ADFGX#Entzifferung

Im zweiten Weltkrieg hatten die Deutschen aus dieser Niederlage gelernt und verwendeten die wesentlich komplexere, maschinelle Ver- und Entschlüsselung durch die Enigma-Maschine. Jedoch führte die Geheimhaltung ihres Aufbaus und damit die Nichtbeachtung von Kerckhoffs' Prinzip dazu, dass die volle mathematische Stärke dieser Konstruktion nicht genutzt wurde. Ab Januar 1940 und nahezu während des gesamten Zweiten Weltkriegs konnte insbesondere das britische Militär weite Teile des deutschen Nachrichtenverkehrs auf allen Ebenen mitlesen. Es wird vermutet, dass dieser Informationsvorsprung den Zweiten Weltkrieg um mindestens zwei Jahre verkürzte oder ihn überhaupt entschied. Heute ist die öffentliche Analyse der Stärken und Schwächen im Aufbau der Enigma Teil eines als exzellent ausgezeichneten Wikipedia-Artikels[48].

### 4.5. Der Advanced Encryption Standard (AES)

Am Beginn dieses Verfahrens stand eine Ausschreibung im Jahr 1997, bei der ein Advanced Encryption Standard[49] (deutsch »fortschrittlicher Verschlüsselungsstandard«) gesucht wurde. Aus fünfzehn eingereichten Vorschlägen wurde in mehreren öffentlichen Konferenzen und Diskussionen ein Algorithmus aus Sieger bestimmt, der überdurchschnittlich schnell ist und, soweit erkennbar, mit realistischem physikalisch-rechnerischem Aufwand nicht gebrochen werden kann. Dieser Algorithmus wurde im Oktober 2000 vom Nationalen Institut für Standards und Technologie der Vereinigten Staaten (NIST) als Standard bekanntgegeben. Der Ablauf dieses Wettbewerbs gilt bis heute als vorbildlich.

Auch bei AES handelt es sich um ein symmetrisches Verschlüsselungsverfahren, das heißt die Schlüssel zum Ver- und Entschlüsseln sind identisch. Der Algorithmus ist eine Blockchiffre, bei der Blöcke von 128 Bit = 16 Byte gemeinsam ver- und entschlüsselt werden. Die möglichen Schlüssellängen sind auf 128, 192 oder 256 Bit festgelegt. Durch zahlreiche Vertauschungen und Verknüpfungen in aufeinander folgenden »Runden« wird die Wahrscheinlichkeit mathematischer Angriffe auf das Verfahren minimiert.

Da zu verschlüsselnde Dokumente praktisch immer länger sind als die Blocklänge von 16 Byte, wird zusätzlich ein Betriebsmodus[50] benötigt, um aufeinander folgende Datenblöcke so miteinander zu verknüpfen, dass daraus nicht Teile des verschlüsselten Inhalts oder der geheime Schlüssel abgeleitet werden können. Sicherer als die Daten erst zu authentifizieren und anschließend zu verschlüsseln ist es, erst zu verschlüsseln und die verschlüsselten Daten anschließend zu authentifizieren.[51] Beide Schlüssel dürfen nicht identisch sein.

### 4.6. Die Festplattenverschlüsselung BitLocker als ein Beispiel für AES

Das Aktivieren der Festplattenverschlüsselung mit BitLocker unter Windows ist einfacher, als es klingt. Doch ist es für Nutzer von Windows Home mit 145 € nicht ganz billig. Getestet habe ich BitLocker unter Windows 10, es funktioniert aber auch unter Windows 8 und Windows 7. BitLocker jetzt noch unter Windows 7 zu installieren, lohnt sich nicht mehr, denn

48 https://de.wikipedia.org/wiki/Enigma_(Maschine)

49 https://de.wikipedia.org/wiki/Advanced_Encryption_Standard

50 https://de.wikipedia.org/wiki/Betriebsmodus_(Kryptographie)

51 https://crypto.stackexchange.com/questions/202/should-we-mac-then-encrypt-or-encrypt-then-mac

ab Januar 2020 werden für dieses Betriebssystem keine aktuellen Sicherheitsupdates mehr ausgeliefert, so dass Windows 7 auf einem Praxisrechner sich spätestens dann verbietet.[52]

Ratsam ist zunächst, alle Daten vom Rechner auf externer Festplatte, SD-Karte oder USB-Stick zu sichern. Bei der Installation von Windows Pro und BitLocker geht selten etwas schief. Aber falls doch, sind dann so keine Daten verloren. Um herauszufinden, welche Edition von Windows auf dem Rechner läuft, kann man zum Beispiel die Windows-Logo-Taste + Pause (neben F12) drücken oder die Windows-Logo-Taste + R drücken und msinfo32 öffnen. Sollte dort Windows Home stehen, muss als Voraussetzung für den Einsatz von BitLocker zunächst das kostenpflichtige Upgrade auf Windows Pro durchgeführt werden.

Für das Upgrade zu Windows 10 Pro wählt man Einstellungen => Update und Sicherheit => Aktivierung => Store aufrufen. Dort klickt man auf die Schaltfläche »Kaufen« und muss sich dann zunächst in ein bestehendes Microsoft-Konto einloggen oder ein neues Konto erstellen. Darin kann zwischen zwei Bezahlmethoden gewählt werden: Sofortüberweisung[53] und Giropay[54]. Den besseren Datenschutz bietet Giropay, jedoch wird diese Methode nicht von allen Banken angeboten. Wer beispielsweise das Praxiskonto bei der apoBank hat, muss auf die Abbuchung von einem Konto bei einer anderen Bank ausweichen oder die Sofortüberweisung nutzen, welche die Zugangsdaten zum Onlinebanking an Dritte weitergibt. Dann sollte man seine Zugangsdaten zum Onlinebanking im Anschluss ändern. Unter »Neue Zahlungsmethode hinzufügen« werden Kreditkarte, PayPal und Klarna angeboten. Sie habe ich an dieser Stelle nicht getestet. Nach erfolgtem Kauf muss der Rechner neu gestartet werden und nach wenigen Minuten ist Windows Pro installiert.

Vor der Aktivierung von BitLocker stellen sich mehrere Glaubens- und Geschmacksfragen. Die erste Frage ist jene für oder wider den in den meisten aktuellen Rechnern verbauten TPM-Chip[55] (Trusted Platform Module). Ob der Rechner über einen TPM-Chip verfügt, findet man heraus, indem man in einem Benutzerkonto mit Administratorrechten die Windows-Logo-Taste + R drückt und tpm.msc öffnet. Dort steht unter »Status«, ob das TPM vorhanden und einsatzbereit ist.

Für dessen Verwendung spricht, dass mit dem Chip die Festplatte nur dann wieder entschlüsselt werden kann, wenn die Hardware des Rechners unverändert ist. Würde man die Festplatte in einen anderen Rechner einbauen, um dort die Verschlüsselung zu brechen, ließe der TPM-Chip den Versuch scheitern. Gegen den TPM-Chip spricht, dass dessen Manipulation durch die NSA oder andere staatliche Stellen nicht auszuschließen ist. Diese könnten dann die Festplatte selbst entschlüsseln oder deren Entschlüsselung durch den Besitzer verhindern.

Nicht zu empfehlen ist die Authentifizierung von BitLocker allein über das TPM, weil einige Angriffe auf die verschlüsselten Daten dann trotzdem möglich wären. So lautet die zweite

52 https://support.microsoft.com/de-de/help/4057281/windows-7-support-will-end-on-january-14-2020

53 https://de.wikipedia.org/wiki/Sofort%C3%BCberweisung

54 https://de.wikipedia.org/wiki/Giropay

55 https://de.wikipedia.org/wiki/Trusted_Platform_Module

Entscheidung, was entweder als alleinige Authentifizierung ohne TPM oder als zweiter Authentifizierungsfaktor zusätzlich zum TPM-Chip gewählt werden soll: Passwort oder Schlüsseldatei?

Vor- und zugleich Nachteil des Passwortes ist, dass es nicht aufgeschrieben werden muss. Was man im Gedächtnis hat, kann zwar nicht ausgespäht, aber vergessen werden. Das Passwort ist weniger komplex als die Schlüsseldatei und deshalb leichter durch Probieren zu finden. Bei jedem Start des Rechners muss es eingegeben werden.

Vor- und zugleich Nachteil der Schlüsseldatei ist deren Länge. Um sich diese Information zu merken, müsste man ein Gedächtniskünstler sein. Deshalb wird die Schlüsseldatei auf einem USB-Stick oder einer SD-Karte abgelegt (die jeweils beliebige weitere, nicht mit BitLocker verschlüsselte Daten enthalten dürfen). Bei jedem Einschalten muss der Datenträger mit der Schlüsseldatei mit dem Rechner verbunden sein und kann dann ggf. bis zum nächsten Einschalten wieder entfernt werden. Auch von der Schlüsseldatei sollte man sich eine oder besser zwei Sicherungskopien anfertigen und sie gut verwahren. Sucht man die Schlüsseldatei auf dem USB-Stick, sieht man zunächst nichts, da sie als Systemdatei gekennzeichnet ist und Windows Systemdateien standardmäßig nicht anzeigt. Das kann man ändern, indem man im Windows-Explorer auf Datei => Ordner- und Suchoptionen ändern => Ansicht geht und das Häkchen bei »Geschützte Systemdateien ausblenden (empfohlen)« entfernt. Jetzt zeigt Windows im Windows-Explorer die Schlüsseldatei mit an, deren Name auf *.BEK endet. Nicht passieren darf es, dass USB-Stick bzw. SD-Karte und damit die Schlüsseldatei gemeinsam mit dem verschlüsselten Rechner in die Hände Unbefugter geraten.

Eine Entscheidungshilfe zu der Frage TPM-Chip, Passwort und/oder Schlüsseldatei bietet dieser Artikel[56] (auf Englisch). Microsoft empfiehlt die Kombination aus TPM-Chip und Passwort. Ich bevorzuge die Kombination aus TPM-Chip und USB-Stick, um mir das Eintippen eines weiteren Passwortes vor dem eigentlichen Windows-Login zu sparen. So macht sich BitLocker beim Benutzer nicht bemerkbar, solange beim Anschalten des Rechners der USB-Stick mit der Schlüsseldatei steckt.

Vor dem Einrichten von BitLocker kann eine dritte Entscheidung getroffen werden. Die Verschlüsselungsstärke von BitLocker ist auf 128 Bit voreingestellt, kann aber auf 256 Bit erhöht werden, ohne dass es die Schnelligkeit der meisten Rechner wesentlich bremst.

Zusätzliche Informationen zu BitLocker findet man im Internet beispielsweise bei bei Microsoft, Wikipedia und YouTube. Um die Installation durchzuführen, müssen wir Windows mitteilen, wie wir uns hinsichtlich TPM-Chip, Password oder Schlüsseldatei und Verschlüsselungsstärke entschieden haben. Das geschieht in den sogenannten Gruppenrichtlinien. Um dorthin zu gelangen, drückt man die Windows-Logo-Taste + R, öffnet gpedit.msc und klickt sich durch bis zum Unterordner Richtlinien für Lokaler Computer => Computerkonfiguration => Administrative Vorlagen => Windows-Komponenten => BitLocker-Laufwerkverschlüsselung. (Bitte nicht in die Benutzerkonfiguration verrutschen, wo es auch Administrative Vorlagen gibt.)

---

56 https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-7/ee706531(v=ws.10)

Im Unterordner BitLocker-Laufwerkverschlüsselung angekommen, stehen auf der rechten Seite mehrere Einträge mit dem Namen »Verschlüsselungsmethode und Verschlüsselungsstärke für Laufwerk auswählen«. Der passende Eintrag ist der für Windows-Version 1511 und höher. Ist man sich nicht sicher, können auch alle Einträge bearbeitet werden. Nach Doppelklick auf den Eintrag muss die Option »Aktiviert« und darunter die gewünschte Kombination aus Verschlüsselungsmethode und Verschlüsselungsstärke gewählt werden, beispielsweise XTS-AES 256-Bit.

Zurück auf der linken Seite wählt man den Unterordner »Betriebssystemlaufwerke« und doppelklickt auf der rechten Seite auf den Eintrag »Zusätzliche Authentifizierung beim Start anfordern«. Auch hier muss wieder die Option »Aktiviert« gewählt werden. Möchte man den TPM-Chip nicht nutzen, muss das Häkchen bei »BitLocker ohne kompatibles TPM zulassen« gesetzt sein. Auch die nachfolgenden Einstellungen zu Passwort (PIN) und Schlüsseldatei können entsprechend den eigenen Wünschen gesetzt werden.

Dieses Fenster kann jetzt wieder geschlossen werden und es geht weiter im Windows-Explorer, wo man mit der rechten Maustaste auf das Betriebssystem-Laufwerk (meist C:) klickt, »Bitlocker aktivieren« wählt und die Laufwerksverschlüsselung startet. Je nach Größe des Laufwerks dauert der Verschlüsselungsvorgang zwischen einer und mehreren Stunden. Dabei ist es ratsam, den ungenutzten Platz des Laufwerks mit zu verschlüsseln, da sich dort Reste gelöschter Daten befinden können, die sonst für einen Angreifer lesbar bleiben. Die Abläufe beim Verschlüsseln der Festplatte zeigt dieses Video[57] (auf Englisch). Am Ende darf nicht vergessen werden, den Wiederherstellungsschlüssel zu sichern, am besten sowohl als Datei auf einem externen Laufwerk als auch als Ausdruck auf Papier.

### 4.7. Eine kurze Geschichte der hybriden Verschlüsselung

Bis in die 1970er Jahre gab es nur symmetrische Verschlüsselung, bei der Sender und Empfänger denselben Schlüssel besitzen mussten. Das wirft das Problem auf, dass es keinen sicheren und einfachen Weg gibt, auf dem der Schlüssel unabhängig von den mit ihm verschlüsselten Informationen vom Sender zum Empfänger gelangen kann. Am sichersten ist es, wenn Sender und Empfänger sich zunächst einmal persönlich treffen, um den gemeinsamen symmetrischen Schlüssel zu vereinbaren.

Nach entweder geheim gehaltenen oder erfolglosen Versuchen am Anfang der 1970er Jahre wurde 1977 von Rivest, Shamir und Adleman das erste sichere asymmetrische Verschlüsselungsverfahren entwickelt und nach ihnen RSA-Verfahren genannt. Bei entsprechend großen Schlüssellängen gilt es bis heute als sicher und wird häufig verwendet. Wenn eine anschauliche Metapher für symmetrische Verschlüsselung die Haustür ist, ist die Metapher für asymmetrische Verschlüsselung der Briefkasten.

Es ist ebenso üblich wie irreführend, von »öffentlichen« und »privaten« asymmetrischen Schlüsseln zu sprechen. Niemand käme auf die Idee, den Briefkastenschlitz als »öffentlichen Briefkastenschlüssel« zu bezeichnen. Während die Funktion eines Briefkastens durch Gesetze der Physik sichergestellt ist, stellen Gesetze der Mathematik sicher, dass es

57 https://www.youtube.com/watch?v=5o9zGAOOg4c

einfach ist, aus einem »privaten Schlüssel« den zugehörigen »öffentlichen Schlüssel« zu berechnen, jedoch äußerst unwahrscheinlich, aus einem »öffentlichen Schlüssel« zurück auf den zugehörigen »privaten Schlüssel« zu schließen.

Ein »öffentlicher Schlüssel« dient entweder dem Zweck, Daten für den Besitzer des zugehörigen »privaten Schlüssels« zu verschlüsseln oder von diesem empfangene Daten auf Authentizität und Integrität zu prüfen, das heißt die Signatur dieser Daten zu überprüfen. Ein »privater Schlüssel« dient entweder dem Zweck, Daten, die mit dem zugehörigen »öffentlichen Schlüssel« verschlüsselt wurden, wieder zu entschlüsseln oder Daten vor dem Versand zu authentisieren und gegen nachträgliche Veränderung zu sichern, das heißt die Daten zu signieren.

Die Prüfung, ob Daten unbeschädigt und deshalb mit hoher Wahrscheinlich authentisch sind, ist unabhängig von Verschlüsselung und funktioniert auch mit im Klartext versandten Daten. Gleichfalls kann man Daten verschlüsseln, ohne sie mittels Signatur vor nachträglicher Veränderung zu schützen. Der Normalfall ist jedoch die gleichzeitige Anwendung von Verschlüsselung und Signatur. Dazu werden zwei Schlüsselpaare benötigt, die beide jeweils aus einem »privaten Schlüssel« und einem »öffentlichen Schlüssel« bestehen.

Sollen Daten verschlüsselt werden, die länger sind, als der asymmetrische Schlüssel selbst, wird asymmetrische Verschlüsselung in der Praxis immer in einer Kombination mit symmetrischer Verschlüsselung verwendet. Die Bezeichnung hierfür ist hybride Verschlüsselung. Asymmetrische Verfahren arbeiten wegen ihres Konstruktionsprinzips wesentlich langsamer als symmetrische Verfahren. Durch diesen Nachteil, dem kein Vorteil gegenüber steht, wurde die theoretische Möglichkeit, aufeinander folgende Blöcke von Daten jeweils mit demselben asymmetrischen Schlüssel zu verschlüsseln, nie standardisiert oder hinreichend auf Schwachstellen geprüft. Gerade bei der Datensicherheit soll man nicht »das Rad neu erfinden«, sondern auf bewährte, schon von vielen Experten durchdachte Konzepte setzen. Reine asymmetrische Verschlüsselung bleibt für Alltagszwecke deshalb bedeutungslos.

Hybride Verschlüsselung verbindet die Schnelligkeit symmetrischer mit dem leichteren Schlüsselaustausch asymmetrischer Verfahren. Zugleich hat sie gegenüber rein symmetrischer Verschlüsselung mehrere Nachteile. Ein Nachteil hybrider Verschlüsselung besteht darin, dass sie gebrochen ist, sobald einer ihrer beiden Teile gebrochen ist. Gelingt es, aus einem öffentlichen Schlüssel den zugehörigen privaten Schlüssel zu berechnen, wird der mit den Daten gemeinsam übertragene symmetrische Schlüssel lesbar. Gelingt es, den symmetrischen Schlüssel durch Ausprobieren zu finden, hat man den asymmetrischen Teil des Verfahrens umgangen.

Ein weiterer Nachteil asymmetrischer und damit auch hybrider Verschlüsselung besteht darin, dass auch sie das eingangs geschilderte Problem nicht vollständig löst, einen sicheren und einfachen Weg zu finden, auf dem ein kryptographischer Schlüssel unabhängig von den mit ihm verschlüsselten Informationen vom Sender zum Empfänger gelangen kann. Da hier Sender und Empfänger sich nicht persönlich treffen, um ihre Schlüssel zu vereinbaren, verbleibt ein Risiko, dass Identitäten gefälscht sind. Während private Schlüssel ihrerseits verschlüsselt oder sicher verwahrt werden müssen, benötigen öffentliche Schlüssel zusätz-

liche Identitätsmerkmale, um Fälschungen vorzubeugen. Das kann die schon länger bestehende Praxis-Homepage sein, die den öffentlichen Schlüssel der Praxis zum Herunterladen anbietet, die telefonische Nachfrage beim (vermeintlichen) Absender einer E-Mail, der ein öffentlicher Schlüssel angehängt ist, vielleicht in Zukunft auch der Überweisungsschein mit Praxisstempel, dem ein öffentlicher Schlüssel in Form eines QR-Code aufgedruckt ist. Für öffentliche Schlüssel muss ein »Netz des Vertrauens«[58] aufgebaut werden.

Alle öffentlichen Schlüssel durch eine zentrale Stelle nicht nur (mit-)speichern, sondern auch beglaubigen zu lassen, führt zunächst in blindes Vertrauen zurück. Denn wer zertifiziert die Zertifizierungsstelle, die das Zertifikat der Zertifizierungsstelle zertifiziert, das...? Die Frage nach dem Vertrauen in zentrale Zertifizierungsstellen für öffentliche Schlüssel ist nicht mathematisch, sondern politisch.

Private Schlüssel gehören, analog Passwörtern, allein in die Hände ihres Besitzers und gehen die Zertifizierungsstelle nichts an. Die Zeitschrift »c't« schrieb in Heft 14/2020[59]: »Seriöse CAs[60] bieten Kunden die Möglichkeit, die Schlüssel im Browser mittels einer anderen Software oder dem etablierten CSR-Verfahren[61] lokal zu generieren.«

Selbst dann noch, wenn dank einer sorgfältig arbeitenden Zertifizierungsstelle keine öffentlichen Schlüssel unter falscher Identität in Umlauf sind, gibt es Angriffsmöglichkeiten, um wahre Identitäten gegen einander auszutauschen. Geht man von der Denkweise symmetrischer Verschlüsselung aus, bei der nur die Identität des einen Schlüssels existiert, ist das zunächst nicht ganz einfach zu verstehen. Wird eine Nachricht erst verschlüsselt und dann signiert, könnte jeder, der diese Nachricht abfängt, die ursprüngliche Signatur entfernen, durch seine eigene Signatur ersetzen und sich so als der wahre Absender ausgeben. Deshalb sollte die Identität des Absenders immer mit verschlüsselt und nicht allein über die Signatur bekannt gegeben werden.

Wird eine Nachricht hingegen erst signiert und dann verschlüsselt, sind die Angriffsmöglichkeiten geringer. Das steht im Gegensatz zu symmetrischer Verschlüsselung, bei der es sicherer ist, erst zu verschlüsseln und dann zu authentifizieren. Als Angreifer kommt hier nur der legitime Empfänger als in Betracht, der die Nachricht entschlüsseln, ihre Signatur beibehalten und beides mit dem öffentlichen Schlüssel eines Dritten neu verschlüsseln könnte. Dieser Angriff wird als »heimliches Weiterleiten« bezeichnet.[62] Deshalb sollte die Identität des Empfängers immer mit signiert und nicht allein über Verschlüsselung sichergestellt werden.

### 4.8. Das RSA-Kryptosystem nach Rivest, Shamir und Adleman

Das RSA-Verfahren nach Rivest, Shamir und Adleman aus dem Jahr 1977 gilt bis heute als sicher und ist ein sehr häufig verwendetes asymmetrisches Verfahren innerhalb hybrider

58 https://de.wikipedia.org/wiki/Web_of_Trust

59 https://www.heise.de/select/ct/2020/14/2014309143965548448

60 https://de.wikipedia.org/wiki/Zertifizierungsstelle

61 https://de.wikipedia.org/wiki/Certificate_Signing_Request

62 https://drpartha.org.in/publications/surreptitious.pdf

Verschlüsselung. Es beruht auf der Mathematik der Primzahlen. Eine Primzahl ist eine ganze Zahl, die größer ist als 1 und die ausschließlich durch sich selbst und durch 1 teilbar ist. Die Folge der Primzahlen beginnt mit 2, 3, 5, 7, 11, 13 und ist unendlich. Es ist wesentlich leichter, zwei Primzahlen miteinander zu multiplizieren als umgekehrt für eine gegebene Zahl auszuprobieren, aus welchen Primfaktoren sie besteht.

Beispielsweise könnte man von der Zahl 143 auf den ersten Blick vermuten, dass sie das Dreifache einer anderen ganzen Zahl sei. Bis man auf die einzig mögliche Primzahlzerlegung in 11 * 13 kommt, dauert es etwas länger. Umgekehrt gibt es eine einfache Rechenvorschrift dafür, die Zahlen 11 und 13 miteinander zu multiplizieren und das Ergebnis 143 zu erhalten.

Auf dieser Grundlage sind zwei geheim gehaltene Primzahlen der »private Schlüssel« des RSA-Verfahrens, der zum Entschlüsseln von Nachrichten dient. Ihr Produkt ist der »öffentliche Schlüssel«, der zum Verschlüsseln verwendet wird und dessen Länge die Schlüssellänge des Verfahrens definiert. Der »öffentliche Schlüssel« darf und soll bekannt gemacht werden, so dass jede(r) dem Besitzer oder der Besitzerin des zugehörigen »privaten Schlüssels« verschlüsselte Nachrichten schreiben kann. Entschlüsseln kann diese Nachricht jedoch nur, wer die beiden Primzahlen kennt.

Die für eine sichere Verschlüsselung verwendeten Primzahlen müssen um ein Vielfaches größer sein als die Zahlen im obigen Beispiel und größer als alle anderen Zahlen in unserem Alltag. Anders als bei symmetrischer Blockverschlüsselung ist die Länge dieser Zahlen frei wählbar. Trotz einiger Gegenargumente[63] nimmt man auch hier meist Längen, die in der binären Darstellung des Computers »runde« Zahlen ergeben.

Grenzstein mit der Zahl $2^{12}$, Südufer des Großen Stechlinsee, Brandenburg

Die voraussichtlich noch für einige Jahre sichere Mindestlänge eines RSA-Schlüssels sind 4096 Bit[64] = 512 Byte. Mit den log(256) / log(10) ≈ 2,40824 Dezimalstellen pro Byte multipliziert und dann aufgerundet entspricht das einer Dezimalzahl mit 1234 Stellen als öffentlichem Schlüssel. Da die Rechenleistung von Supercomputern auch ohne Quantenkryptographie weiter steigt, müsste man für langfristig sicher aufzubewahrende Daten, wie Gesundheitsdaten es sind, »schon heute Schlüssel mit 15360 Bit nutzen.«[65] Denn es nützt nichts, wenn wir die Daten mit heute sicheren Schlüsseln von 2048 oder 4096 Bit Länge übertragen, sie dabei aber möglicherweise gespeichert und so lange aufgehoben werden, bis Schlüssel dieser Länge

63 https://blog.josefsson.org/2016/11/03/why-i-dont-use-2048-or-4096-rsa-key-sizes/

64 https://www.golem.de/news/verschluesselung-was-noch-sicher-ist-1309-101457-5.html

65 https://www.heise.de/security/artikel/Kryptographie-in-der-IT-Empfehlungen-zu-Verschluesselung-und-Verfahren-3221002.html?seite=4

gebrochen werden können.[66] Laut einer Analyse des Nationalen Instituts für Standards und Technologie der Vereinigten Staaten (NIST)[67] gilt, dass auf herkömmlichen Computern und außerhalb der Quantenkryptographie erst bei 15360 Bit Schlüssellänge ein ähnlich großer Rechenaufwand zum Brechen einer RSA-Verschlüsselung geleistet werden muss, wie er zum Brechen einer AES-256-Verschlüsselung erforderlich wäre.

Nicht nur beim Ver- und Entschlüsseln wird das RSA-Verfahren angewandt, sondern auch bei digitalen Signaturen. Schon ihrem Namen nach ist die digitale Signatur so etwas wie eine Unterschrift: nur eine bestimmte Person (oder ein bestimmtes Gerät) kann sie erzeugen und alle können sie überprüfen. Damit erfüllt die digitale Signatur den doppelten Zweck, die Authentizität von Daten (d.h. die sichere Zuordnung zu einem bekannten Absender) und deren Integrität (d.h. ihre Unversehrtheit seit dem Absenden) sicherzustellen. Hierin unterscheidet die digitale Signatur sich von Verschlüsselung, die Vertraulichkeit herstellen soll.

Es wäre ein schönes Wortspiel zu behaupten, eine Signatur erzeuge Vertrauen und eine Verschlüsselung erzeuge Vertraulichkeit. Aber das stimmt nicht. Dass auf der Basis digitaler Signaturen demnächst Freundschaften geschlossen werden, ist nicht zu erwarten. Vielmehr ist ein typisches Gedankenexperiment über Signaturen ein eher kriegerisches: das Problem der byzantinischen Generäle[68].

Grob gesagt entsteht eine digitale Signatur beim »Entschlüsseln« einer Zahl, die gar nicht verschlüsselt war. Zumindest ist das so bei der häufig verwendeten Signatur mittels des RSA-Kryptosystems. Das mag unlogisch klingen, jedoch sind »Entschlüsseln« und »Signieren« nur nachträgliche menschliche Deutungen für die jeweils gleichen mathematischen Operation des modularen Potenzierens. Dem Computer ist es egal, was die Zahlen bedeuten, mit denen er rechnet. Der Philosoph Jean Baudrillard spricht von der Mathematik als referenzlosen Zeichen[69]. Diese referenzlosen Zeichen treten als »Hyperrealität« in Konkurrenz mit Konstrukten, die wir bisher als unsere »Realität« wahrgenommen haben.

Wer die nötige Geduld mitbringt, kann sich in Kapitel 5 der »Standards für asymmetrische Kryptographie«[70] von den identischen Formeln für das Signieren (Kapitel 5.2.1.) und das Entschlüsseln (Kapitel 5.1.2.) überzeugen. Genau wie das asymmetrische Entschlüsseln benötigt auch das digitale Signieren einen privaten Schlüssel. Und genau wie das asymmetrische Verschlüsseln benötigt auch das Überprüfen einer digitalen Signatur einen öffentlichen Schlüssel. Ein von Zeit zu Zeit diskutiertes Recht auf Verschlüsselung[71] müsste deshalb auch ein Recht auf digitale Signatur enthalten.

66 https://news.netcraft.com/archives/2013/06/25/ssl-intercepted-today-decrypted-tomorrow.html

67 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

68 https://de.wikipedia.org/wiki/Byzantinischer_Fehler

69 https://de.wikipedia.org/wiki/Der_symbolische_Tausch_und_der_Tod

70 https://tools.ietf.org/html/rfc8017

71 https://www.bundestag.de/dokumente/textarchiv/2020/kw05-pa-inneres-669564

Beim digitalen Signieren wird zunächst der komplette Inhalt eines Dokuments mit einer kryptographischen Hashfunktion[72] wie SHA-512 in eine Zahl mit vorgegebener Länge umgerechnet. Erst anschließend wird dieser Hashwert von einem bestimmten Absender für beliebige Empfänger »sicher entpackt« – um die Metapher des »Entschlüsselns« einer Zahl, die nicht verschlüsselt war, hier noch einmal umzuformulieren.

Beim Prüfen einer digitalen Signatur durch einen beliebigen Empfänger wird der Hashwert des fraglichen Dokuments berechnet und mit dem öffentlichen Schlüssel des bekannten Absenders »verschlüsselt«. Stimmt das Ergebnis mit der übermittelten Signatur des Dokuments überein, gelten Authentizität und Integrität des Dokuments als bestätigt.

Wer jetzt sagt, dann hätte der Absender sich das »sichere Entpacken« des Hashwertes in eine Signatur doch sparen können und hätte den von jedem leicht zu berechnenden Hashwert lieber unverändert oder besser gar nicht mitgeschickt, ist auf die Metapher von »Klartext« und »Schlüsseltext« hereingefallen, die sich hier mit der Metapher von »Verschlüsseln« und »Entschlüsseln« überkreuzt. Die Kunst von Maurits Cornelis Escher[73] verdeutlicht den Unterschied zwischen einem mathematisch-geometrischen Muster und dessen Interpretation.

Um es zusammenzufassen: In der Praxis wird das RSA-Kryptosystem nach Rivest, Shamir und Adleman nie direkt auf die eigentlichen Daten angewandt. Nur das Ergebnis einer Hashfunktion wird signiert bzw. verifiziert und nur ein symmetrischer Schlüssel wird asymmetrisch verschlüsselt bzw. entschlüsselt. Die Mathematik hinter allen diesen Vorgängen ist im Wesentlichen dieselbe.

### 4.9. Das Kryptomodul der KBV als ein Beispiel für hybride Verschlüsselung

Eine Anwendung, die das Prinzip hybrider Verschlüsselung verdeutlicht, ist das Kryptomodul der Kassenärztlichen Bundesvereinigung[74]. Zunächst mag es wie ein ausschließlich asymmetrisches Verfahren erscheinen, da Benutzer die symmetrischen Schlüssel nie zu Gesicht bekommen. Wer mit der Kassenärztlichen Vereinigung abrechnet, schickt dorthin am Ende eines Quartals eine Datei mit den Abrechnungsdaten. Der Name dieser Datei endet auf .CON.XKM und man kann sie entweder auf eine CD brennen und per Einschreiben an seine Geschäftsstelle der Kassenärztlichen Vereinigung schicken oder über ein sogenanntes sicheres Netz wie die Telematikinfrastruktur und das KV-SafeNet elektronisch einreichen.

Öffnet man diese Datei mit einem Texteditor, entdeckt man überwiegend unleserliche Zeichenfolgen. Das sind die Abrechnungsdaten, die symmetrisch per AES und einem 128 Bit langen Schlüssel verschlüsselt wurden. Sie stammen aus einer anderen Datei mit der Endung .CON, welche die Quartalsabrechnung im Klartext enthält und nicht mit eingereicht werden darf.

---

72 https://de.wikipedia.org/wiki/Kryptographische_Hashfunktion

73 https://www.wikiart.org/de/m-c-escher/belvedere

74 ftp://ftp.kbv.de/ita-update/KBV-Software/Kryptomodul/

Den verschlüsselten Daten vorangestellt sind einige Verwaltungsdaten wie Betriebsstättennummer und Erstellungsdatum im Klartext, ein SHA-256-Hashwert[75] über die unverschlüsselten Daten zur Prüfung auf nachträgliche Beschädigungen und der einmalige, zufällig erzeugte symmetrische 128-Bit-Schlüssel. Da es zwecklos wäre, Daten zu verschlüsseln, wenn man ihren Schlüssel offen lesbar mitliefert, ist dieser per RSA-Verschlüsselung und dem 2048 Bit langen öffentlichen Schlüssel der Kassenärztlichen Bundesvereinigung verschlüsselt. Dadurch nimmt er in der Datei nicht mehr 128 Bit, sondern 2048 Bit = 256 Byte ein. Die verschlüsselte Datei .CON.XKM ist also stets länger als die unverschlüsselte Datei .CON.

Da diese Datei nicht vom Absender signiert ist, sondern nur gegen Beschädigungen ab dem Zeitpunkt ihrer Verschlüsselung gesichert, wäre es theoretisch möglich, dass ein Angreifer die Datei Datei .CON.XKM bei ihrer elektronischen Übertragung oder auf dem Postweg abfängt. Dann wären die originalen Daten zwar nicht zu entschlüsseln, jedoch könnten an ihrer Stelle komplett andere Daten mit dem öffentlichen Schlüssel der Kassenärztlichen Bundesvereinigung verschlüsselt, ihrerseits gegen nachträgliche Beschädigungen gesichert und bei der Kassenärztlichen Vereinigung zur Abrechnung im Namen der absendenden Praxis eingereicht werden.

### 4.10. Andere dezentrale elektronische Patientenakten

In den Jahren 2012 bis 2014 wurde das Projekt maxiDoc[76] durch eine gleichnamige GmbH entwickelt. Testpatienten erhielten USB-Sticks, auf denen patientenbezogene Daten verschlüsselt werden konnten. Das dabei genutzte Verschlüsselungsverfahren bleibt im Dunkeln, jedoch klärt die Diplomarbeit von Monika C.G. Stübbecke mit dem Titel »Die elektronische Gesundheitskarte und das Problem des Datenschutzes im Bereich medizinischer Leistung«[77] über einige der Merkmale von maxiDoc auf. So »...ist eine Passworteingabe durch den jeweiligen Patienten nicht von Nöten, wenn der behandelnde Arzt über eine maxiDoc-Schreibsoftware verfügt. Auch die Speicherung von personenbezogenen Gesundheitsdaten auf den maxiDoc-Stick ist aufgrund der maxiDoc-Schreibsoftware ohne Passworteingabe durch den Patienten möglich. Sollte das Passwort jedoch vergessen oder verlegt werden, kann dieses von dem behandelnden Arzt auf das Standardpasswort zurückgesetzt werden und der Zugriff auf die personenbezogenen Gesundheitsdaten des jeweiligen Patienten ist wieder möglich.«

Mit anderen Worten ist die maxiDoc-Arztsoftware selbst das zu schützende Geheimnis, nicht so sehr das Passwort des Patienten. Sollte die Software von Hackern ins Internet gestellt werden, müsste in allen teilnehmenden Praxen die Software erneuert und anschließend alle Sticks neu verschlüsselt werden. Über differenzierte Zugriffsrechte verfügt der maxiDoc-Stick laut der Arbeit von Frau Stübbecke nicht. Jedoch kann ein Patient Dateien, die nicht von Behandlern eingesehen werden sollen, in einen separaten Container auf dem Stick verschieben.

---

75 https://de.wikipedia.org/wiki/SHA-256

76 https://egesundheit.nrw.de/projekt/maxidoc/

77 https://books.google.de/books?id=geNoAQAAQBAJ

Die Defizite von maxiDoc gegenüber dem Wissensstand der Kryptographie können als Hintergrund dienen, vor dem sich eine dezentrale und möglichst sichere elektronische Patientenakte entwerfen lässt. Dabei kommt man meiner Meinung nach um vier Dinge nicht herum: 1. hybride Verschlüsselung, 2. Meiden digitaler Vernetzung, 3. Quelloffenheit und 4. »Wirtschaftsförderung«. In der Diskussion um die Zwangsvernetzung des deutschen Gesundheitswesens in der Telematikinfrastruktur besteht die Gefahr, dass Begriffe wie »Digitalisierung« und »Wirtschaftsförderung durch die Politik« zunehmend negativ besetzt werden. Dabei wäre ich einverstanden, wenn Digitalisierung zu spürbaren Verbesserungen bei meiner Arbeit führen würde, nicht nur zu dem jetzigen Gegensatz zwischen spürbarer Mehrbelastung und beworbenen, vielleicht in Zukunft möglichen Verbesserungen. Weiterhin fände ich es gut, wenn in der Konkurrenz zwischen dem größeren Konzern und der besseren Idee nicht der größere Konzern gewönne.

Eine weitere dezentrale elektronische Patientenakte wurde durch Dieter Adler vom Deutschen Psychotherapeutennetzwerk entworfen. Sie besteht aus einer herkömmlichen elektronischen Gesundheitskarte mit ausklappbarem USB-Anschluss. Wie ein hardwareverschlüsselter USB-Stick speichert sie Daten in verschlüsselter Form. Versicherte können verschlüsselte Sicherungskopien von diesen Daten auf ihrem Computer ablegen. Durch die Verschlüsselung sind die Gesundheitsdaten auch dann noch vor dem Zugriff Fremder geschützt, wenn Karte oder Computer verloren gehen. So können Versicherte ihre elektronische Patientenakte in der Brieftasche bei sich tragen. Durch ihre Form unterscheidet die Karte sich von zahlreichen anderen USB-Sticks, die man wahrscheinlich außerdem besitzt.

Vorbehaltlich einer genaueren Darstellung durch Herrn Adler erinnert mich seine Lösung an den Nitrokey[78], nur eben in Form einer Scheckkarte. An das Projekt von Dieter Adler und an den Nitrokey schließen sich meine Gedanken für eine dezentrale elektronische Patientenakte an.

Eine elektronische Patientenakte symmetrisch zu verschlüsseln verbietet sich nicht nur für eine zentralisierte Cloud-Lösung wie die Telematikinfrastruktur. Das gilt auch dann, wenn die Daten in der Hand der Versicherten bleiben. Bei symmetrischer Verschlüsselung geschieht das Schreiben (Verschlüsseln) und das Lesen (Entschlüsseln) mit demselben Schlüssel. Wie ließe sich dann verhindern, dass Patienten ihre verschlüsselten Befunde nachträglich ändern oder sie gleich selbst schreiben? Vielleicht, indem Befunde nicht mit dem Schlüssel des Versicherten, sondern dem des Behandlers verschlüsselt werden? Dann könnten Versicherte ihre Befunde nur in Gegenwart des Behandlers lesen. Außerdem müssten sendender und empfangender Behandler zuvor einen gemeinsamen geheimen Schlüssel vereinbart haben, der wiederum bewirkt, dass der eine im Namen des anderen Befunde schreiben könnte.

Demnach benötigt die elektronische Patientenakte hybride Verschlüsselung. Wie hybride Verschlüsselung genau funktioniert, steht in Kapitel »4.7. Eine kurze Geschichte der hybriden Verschlüsselung«. Noch genauer zeigt das der Quellkode meines Programms.

78 https://www.nitrokey.com

Bei hybrider Verschlüsselung besitzt ein Teilnehmer bzw. eine Teilnehmerin immer zwei Schlüssel: einen geheimen (privaten) Schlüssel und einen öffentlichen Schlüssel. (Genau genommen ist der öffentliche Schlüssel eine Teilinformation aus dem privaten Schlüssel.) Mit dem privaten Schlüssel können Daten gelesen (entschlüsselt) und mit einer Identität versehen (signiert) werden. Mit dem öffentlichen Schlüssel können Daten geschrieben (verschlüsselt) und die Identität eines Absenders geprüft werden.

Hybrid verschlüsselte Daten schickt man am besten nicht durch das Internet. Noch ist diese Art der Verschlüsselung sicher. Durch die fortschreitende Entwicklung der Quantencomputer wird sich das in den nächsten 20 Jahren voraussichtlich drastisch ändern. Heute als »sicher« verschickte Daten können so lange gespeichert werden, bis sie aufgrund besserer Entschlüsselungstechnik nicht mehr sicher sind. Wer seine Onlinekommunikation auf das »Pferd« hybride Verschlüsselung setzt (also wir alle), sollte sie nicht für geschenkt nehmen und sollte wissen, dass hybride Verschlüsselung rasch altert.

Was hybride Verschlüsselung für eine dezentrale elektronische Patientenakte bedeutet, werde ich im Folgenden beschreiben. Zur Einfachheit werde ich dabei den absendenden Behandler bzw. die absendende Behandlerin mit A bezeichnen, Patientin oder Patient mit P und die empfangende Behandlerin bzw. den empfangenden Behandler mit E.

Um Daten in eine elektronische Patientenakte zu schreiben, hängt A zunächst den öffentlichen Schlüssel von P an den noch unverschlüsselten Befund. Das macht eine Sicherheitslücke unmöglich, die als »heimliches Weiterleiten« bezeichnet wird. Dann signiert A beides mit dem eigenen geheimen Schlüssel, verschlüsselt beides mit dem öffentlichen Schlüssel von P und hängt sowohl die erzeugte Signatur als auch den eigenen öffentlichen Schlüssel an die Datei. Das Ergebnis wird in die elektronische Patientenakte übertragen.

Ohne den ersten Schritt mit dem öffentlichen Schlüssel von P könnte P nun den Befund mit dem eigenen privaten Schlüssel entschlüsseln, die Signatur und den öffentlichen Schlüssel von A beibehalten, den unverschlüsselten Befund mit dem öffentlichen Schlüssel einer Person X neu verschlüsseln und anschließend in die Patientenakte von X schleusen. Dann sähe es so aus, als habe A seine Schweigepflicht gebrochen, indem er den Befund von P in die Patientenakte von X geschrieben hat. Wird jedoch der öffentliche Schlüssel von P durch A mit signiert, ist der Beweis erbracht, dass P der wirkliche Adressat war. Das gilt unter der Bedingung, dass P sich nicht den geheimen Schlüssel von A verschafft hat, um in dessen Namen Signaturen neu zu erstellen. Eine spannende Frage ist, ob und wie in der Telematikinfrastruktur heimliches Weiterleiten verhindert wird.

Hat A einen Befund gegen heimliches Weiterleiten gesichert, signiert, verschlüsselt und auf dem USB-Datenträger in den Händen von P gespeichert, kann P damit zu Behandler E gehen. Zwischenzeitlich unbemerkt ändern und anschließend neu verschlüsseln kann P den Befund nicht. Das würde zu einer Fehlermeldung führen, wenn E die Signatur der Daten anhand des angehängten öffentlichen Schlüssels von A überprüft. Um diese Meldung zu umgehen, bräuchte P wiederum den geheimen Schlüssel von A.

Bei E angekommen, scheint sich für P die Alles-Oder-Nichts-Frage zu stellen, ob P seinen geheimen Schlüssel an E weitergibt, so dass E in der Lage ist, den von A verschlüsselten

Befund zu lesen. Wenn aber E den geheimen Schlüssel von P besitzt, könnte E sich nicht nur als P ausgeben, sondern könnte alle anderen Befunde entschlüsseln, die in der Patientenakte gespeichert sind. Falls E Radiologe und A Hausarzt ist, könnte der Radiologe nicht nur den Befund des Hausarztes lesen, sondern beispielsweise auch den der Dermatologin an den Chirurgen und den des Psychotherapeuten an die Psychiaterin. Für dieses Problem gibt es mindestens zwei Lösungen: eine, die am privaten Computer von P stattfindet und eine, die einen zwei- bis dreistelligen Eurobetrag kostet.

Die erste Lösung ist, dass P schon zu Hause entscheidet, welche Befunde E erhalten soll. Dazu lädt P den öffentlichen Schlüssel von E von dessen Praxis-Homepage herunter, entschlüsselt die gewünschten Befunde mit dem eigenen privaten Schlüssel und verschlüsselt sie erneut in seiner elektronischen Patientenakte mit dem öffentlichen Schlüssel von E. Die von A erstellte Signatur hängt P dabei an das Dokument, um nachzuweisen, dass es beim Umschlüsseln nicht verändert wurde.

Die zweite Lösung ist, dass der Entschlüsselungsvorgang auf der Gesundheitskarte von P stattfindet, so dass der darin befindliche private Schlüssel die Karte nicht verlässt. Diese Funktion bietet beispielsweise der Nitrokey. Verschlüsselung muss die Karte nicht beherrschen, denn hybride Verschlüsselung funktioniert mit öffentlichen Schlüsseln und kann deshalb auf den Computern der Behandler laufen.

Zurück in das Sprechzimmer von E, wo P entweder mit der zur Entschlüsselung fähigen Gesundheitskarte oder mit den zuvor für E umgeschlüsselten Befunden sitzt. In dem von Dieter Adler entwickelten Modell wird nun P von E gebeten, sich vor einen Bildschirm zu setzen, den nur P einsehen kann. Die Speicherkarte wird in den USB-Anschluss des Computers von E geschoben.

Der Computer von E schaut zunächst in einer Liste nach, die regelmäßig von den Krankenkassen herausgegeben wird. In dieser Liste sind die öffentlichen Schlüssel aller Versicherten verzeichnet, mit einem zusammen mit einem Salt[79] berechneten Hash-Kode (einer gegen Ausspähen gesicherten Prüfsumme) über Name, Geburtsdatum, Adresse und Versichertennummer. Ist P in dieser Liste nicht zu finden, steht P vor einer Wahl. Entweder geht P persönlich zur Krankenkasse, um einen aktuellen Versicherungsnachweis auf die Karte mit der Patientenakte schreiben zu lassen oder P verbindet den USB-Anschluss der Karte mit einem zweiten Computer von E, der mit dem Internet verbunden ist. Dann kann online der Abgleich der Stammdaten und des Versichertennachweises mit der Krankenkasse von P stattfinden.

In den meisten Fällen jedoch sind öffentlicher Schlüssel, Name und Adresse von P dieselben geblieben, so dass der Abgleich mit der Liste der bei den Krankenkassen registrierten öffentlichen Schlüssel positiv verläuft. Nun zeigt der Bildschirm, den nur P einsehen kann, die Namen von allen in der Gesundheitsakte enthaltenen Dateien an. Die Dateinamen könnte E in seinem Computer heimlich protokollieren und daraus Rückschlüsse über andere Behandler ziehen, die zuvor Daten in die Patientenakte von P geschrieben haben. Ist ein

79 https://de.wikipedia.org/wiki/Salt_(Kryptologie)

Patient mit sehr vielen Einträgen Chroniker oder Ärztehopper? Lautet einer der Dateinamen »HIV-Befund« oder enthält er den Kode »F60«?

Hier ist es die Aufgabe von P, sich gegen das Risiko abzusichern, dass E heimlich die Metadaten[80] der elektronischen Patientenakte ausliest. Dazu gibt es mindestens drei Möglichkeiten: einen symmetrisch verschlüsselten Container auf der Karte wie bei maxiDoc, einen verborgenen, hardwareverschlüsselten Container wie im Nitrokey Storage 2 oder die Nutzung mehrerer Gesundheitskarten. Das könnte beispielsweise eine erste Gesundheitskarte für Chirurgie, Dermatologie und Radiologie sein, eine zweite für Psychiatrie und Psychotherapie und eine dritte für den Zahnarzt. Da die Krankenkasse ohnehin Fotos druckt, wird sie gern eine Motivauswahl anbieten, die verschiedene Speicherkarten leicht voneinander unterscheidbar macht.

Um bestimmte Befunde für E freizugeben, hat P entweder diese Arbeit schon zu Hause durch das Umschlüsseln erledigt. Oder P verfügt über die Gesundheitskarte mit Entschlüsselungsfunktion. Dann muss P für jeden Befund, der für E freigegeben werden soll, zweimal mit einer Maus klicken. Das erste Mal, um den für E zu entschlüsselnden Befund zu markieren und das zweite Mal, um einen Entschlüsselungsvorgang auf der Gesundheitskarte freizugeben. Die Gesundheitskarte hat einen internen Zähler durchgeführter Entschlüsselungsvorgänge, der auf dem Bildschirm von E angezeigt wird und außerdem von P zu Hause am USB-Anschluss des eigenen Computers überprüft werden kann.

Zum Verständnis: »Entschlüsselungsvorgang« bedeutet hier das Umrechnen eines asymmetrisch verschlüsselten symmetrischen Schlüssels in einen symmetrischen Schlüssel im Klartext. Wegen der Abfolge aus erst symmetrischer Verschlüsselung und dann asymmetrischer Verschlüsselung des symmetrischen Schlüssels beim Schreiben sowie der asymmetrischen Entschlüsselung des symmetrischen Schlüssels, gefolgt von symmetrischer Entschlüsselung beim Lesen, spricht man von einem hybriden Verfahren.

Konkret könnte das heißen, dass der Computer von A einen zufälligen 256-Bit-Wert berechnet hat, mit diesem hat er dann im AES-Verfahren den Befund von P verschlüsselt. Dieser 256-Bit-Wert wurde anschließend auf dem Computer von A mit dem öffentlichen 4096-Bit-Schlüssel von P im RSA-Verfahren verschlüsselt, was wiederum 4096 Bit ergab, allerdings mit anderer Verteilung von Nullen und Einsen innerhalb der 4096 Bit. Diese 4096 Ergebnisbits wurden gemeinsam mit dem Befund auf der Gesundheitskarte von P gespeichert. Von dort liest sie nun der Computer von E und schickt sie zurück an die Gesundheitskarte zum Entschlüsseln. Die berechnet mit Hilfe des auf ihr gespeicherten geheimen Schlüssels den 256-Bit-Wert und schickt ihn zurück an den Computer von E. Mit diesen 256 Bit entschlüsselt der Computer von E nun nach dem AES-Verfahren den gewünschten Befund und legt ihn auf dem Computer von E ab. Es muss also nicht befürchtet werden, dass beim Entschlüsseln eines besonders langen Befundes »die Gesundheitskarte heiß läuft«. Sie bearbeitet pro Befund maximal 4096 Bit. Weil der 256-Bit-Wert auf dem Computer von A zufällig erzeugt wurde, ist es nicht möglich, damit weitere Befunde aus dieser Patientenakte (oder aus irgend einer anderen) zu entschlüsseln.

80 https://de.wikipedia.org/wiki/Metadaten

Eine Angriffsmöglichkeit besteht darin, dass A die symmetrischen 256-Bit-Schlüssel nicht zufällig erzeugt, sondern für alle Befunde denselben Schlüssel verwendet. Das würde den asymmetrischen Teil des Verfahrens umgehen und eine Person, die diesen symmetrischen Schlüssel von A mitgeteilt bekäme, könnte alle von A geschriebenen Befunde lesen. Ein gewisser Schutz hiergegen wäre es, wenn jeder Behandler, der von einem Patienten die Erlaubnis erhält, mehrere Befunde zu entschlüsseln, durch seine Software automatisch überprüfen ließe, ob dabei identische symmetrische Schlüssel auftreten. Diese Schlüssel dürfen jedoch nicht über den einen Lesevorgang hinaus gespeichert werden, dann würden neue Angriffsmöglichkeiten entstehen. Insgesamt ist das Risiko für einen solchen Angriff als gering einzuschätzen. Wer Befunde Unbefugten zugänglich machen wollte, hätte direktere Wege zur Verfügung als eine Manipulation in der Patientenakte.

Selbstverständlich wird ein Zugriffsprotokoll in die Patientenakte geschrieben und auf dieselbe Weise mit dem privaten Schlüssel von E signiert und mit dem öffentlichen Schlüssel von P verschlüsselt, wie ein Befund. Löschungen hingegen dürfen keine Spuren hinterlassen, müssen also das Zugriffsprotokoll des zugehörigen Schreibvorgangs mit löschen. Ein Eintrag »hier wurde ein psychiatrischer Befund gelöscht« wäre keine Löschung.

Bei der Kommunikation zwischen Behandlern (also zwischen A und E, ohne P) wäre es unpraktisch, eine Karte mit USB-Anschluss herumzureichen oder sogar mit der Post zu verschicken. Hier muss der Datenaustausch über das Internet laufen. Ein erster Schritt wäre, wenn mehr Kolleginnen und Kollegen sich OpenPGP einrichten würden. Jedoch ist selbst dort Vorsicht vor böswilligen Kollegen geboten, denn OpenPGP lässt sich nicht gegen heimliches Weiterleiten[81] sichern. Zudem ist auch OpenPGP ein hybrides Verfahren.

Am widerstandsfähigsten gegen Quantenkryptopgraphie sind symmetrische Verfahren wie der Advanced Encryption Standard. Behandler, die häufig miteinander kommunizieren und die außerdem in räumlicher Nähe zu einander tätig sind, könnten sich einmal treffen, um gemeinsame symmetrische Schlüssel für ihre Internetkommunikation zu vereinbaren. Mittelfristig könnte ein Kollegennetzwerk zu dem Zweck eine quelloffene und alltagstaugliche Software zur Schlüsselvereinbarung unter Nutzung der Supersingular isogeny Diffie–Hellman key exchange (SIDH)[82] entwickeln. Dieser Algorithmus ist durch Quantenkryptographie nicht angreifbar.

Der Versuch, hybride Verschlüsselungslösungen zu finden, die auf Primzahlenfaktorisierung beruhen, aber sicherer sind als OpenPGP, beispielsweise mit meinem Programm OpenDear und / oder mit dem Nitrokey werden vorübergehende Lösungen bleiben. Vielleicht werden sie einige Monate oder Jahre länger gegen den Fortschritt der Quantencomputer Bestand haben als OpenPGP, vielleicht auch nicht. Die kurze Antwort auf die Frage nach sicherer interkollegialer Kommunikation ist deshalb: sich zur Vereinbarung symmetrischer Verschlüsselung einmal persönlich treffen oder weiter Briefe schreiben. Im Spätherbst 1812 verließ Napoleon Bonaparte mit seinen Truppen Moskau[83]. Unzuverlässigen Quellen zufolge soll er dabei gesagt haben: »Wer zu spät geht, den bestraft das Leben«.

---

81 http://world.std.com/~dtd/sign_encrypt/sign_encrypt7.html

82 https://github.com/Microsoft/PQCrypto-SIDH

83 https://www.bbc.co.uk/programmes/m0008jd2

### 4.11. Quantencomputer

Auch schon in heutiger Elektronik werden quantenmechanische Zustände genutzt, jedoch nur zum Speichen von Informationen. So funktionieren die in USB-Sticks verbauten Flash-Speicher über den quantenmechanischen Tunneleffekt[84]. Will man die Gesetze der Quantenmechanik zur Informationsverarbeitung nutzen, erhält man anstelle von Bits, die als Zustand entweder 0 oder 1 annehmen können, sogenannte Qubits[85], in denen die beiden Zustände sich zeitgleich überlagern.

Prinzipiell können Quantencomputer keine neuen Probleme lösen, einige Probleme werden sie allerdings deutlich schneller lösen können als herkömmliche Rechentechnik. Durch die Überlagerung von Verschränkung von Quantenzuständen können Suchprozesse effizienter gestaltet werden. Und das Brechen kryptographischer Schlüssel besteht gerade in der Suche nach dem einen Schlüssel unter vielen, der zu interpretierbaren Daten führt. Auch das Ermitteln von Primzahlen, deren Multiplikation einen bekannten öffentlichen Schlüssel ergibt, ähnelt einem Suchprozess.

Der gegenwärtige Stand der Technik ist offenbar der Quantenprozessor »Sycamore«[86] von Google, der mit 53 Qubits gleichzeitig rechnen kann. Jener Artikel der FAZ macht die folgende Rechnung auf: Um einen heute üblichen asymmetrischen Schlüssel mit 2048 Bits in seine Primfaktoren zu zerlegen, bräuchte ein Quantencomputer aus 10 Millionen Qubits 16 Stunden und würde dabei eine Leistung ähnlich der Antriebsleistung eines Hochgeschwindigkeitszugs benötigen. Leider fehlt eine Quelle für diese Rechnung und ein Kommentator unter dem Artikel bemerkt richtig die physikalische Unmöglichkeit, dass sich das angeblich auf der Fläche eines Quadratzentimeters realisieren ließe.

Naturwissenschaftlich besser fundiert erscheint mir ein Artikel[87], in dem berechnet wird, dass 2953 Qubits erforderlich wären, um einen AES-Schlüssel mit einer Länge von 128 Bit zu brechen. Zum Durchprobieren aller AES-Schlüssel mit einer Länge von 256 Bit würden 6681 Qubits benötigt. Leider ist dort der dafür benötigte Zeit- und Energieaufwand nicht unmittelbar ersichtlich.

Ein weiteres Puzzlestück liefert der Artikel »Post-quantum RSA«[88]. Quantenkryptographische Faktorisierungsverfahren wie der Shor-Algorithmus arbeiten ungleich schneller als quantenkryptographische Suchverfahren wie der Grover-Algorithmus. Das ist an der sogenannten O-Notation für deren Laufzeit zu erkennen. An der Stelle, wo beim Shor-Algorithmus[89] der natürliche Logarithmus steht, steht beim Grover-Algorithmus[90] die Quadratwurzel. Große Zahlen werden durch Logarithmieren wesentlich »kleiner gemacht«

84 https://de.wikipedia.org/wiki/Tunneleffekt#Flash-Speicher

85 https://de.wikipedia.org/wiki/Qubit

86 https://www.faz.net/aktuell/wissen/computer-mathematik/google-das-neue-zeitalter-mit-dem-quantencomputer-16452509.html

87 https://arxiv.org/pdf/1512.04965.pdf

88 https://cr.yp.to/papers/pqrsa-20170419.pdf

89 https://de.wikipedia.org/wiki/Shor-Algorithmus#Eigenschaften

90 https://de.wikipedia.org/wiki/Grover-Algorithmus

als durch Wurzelziehen. So hat die Quadratwurzel aus einhundert Milliarden sechs Stellen vor dem Komma, aber der natürliche Logarithmus aus einhundert Milliarden ist rund 25. Dass in der O-Notation des Shor-Algorithmus außerdem noch die dritte Potenz steht, ändert daran kaum etwas. Auch $25^3$=15625 verfehlt sechs Stellen vor dem Komma.

Konsens scheint zu sein, dass die unteren bis mittleren Größen heute noch sicherer kryptographischer Schlüssel, darunter AES-128 und RSA-2048, in den nächsten Jahren oder Jahrzehnten durch Fortschritte herkömmlicher Rechentechnik und durch Fortschritte in der Quantenkryptographie unsicher gemacht werden könnten. Tritt diese Unsicherheit ein, wird durch jeden weiteren technischen Fortschritt der Quantencomputer die Sicherheit der Schlüssel bei asymmetrischen Verfahren wie RSA logarithmisch zur Anzahl möglicher Werte »abgeschmolzen«, bei symmetrischen Verfahren wie AES nur mit der Quadratwurzel. Für sichere asymmetrische Schlüssel käme man irgendwann bei einer Länge von Giga- und Terabyte an, die länger wäre als alle Daten, die jemals damit geschützt werden.

Auch dann noch werden sichere symmetrische Schlüssel in einen QR-Code passen. Ähnlich unproblematisch ist Quantenkryptographie für die zur Nachrichtenauthentifizierung verwendeten Hash-Funktionen. Zudem wird an kryptographischen Verfahren geforscht, die voraussichtlich in der Lage sein werden, auch den Rechenleistungen der Quantencomputer zu widerstehen.

Aus diesen Gründen wäre es eine unzulässige Verallgemeinerung zu behaupten, dass sich künftig die Zustände »Vertraulichkeit bewahrt« und »Vertraulichkeit gebrochen« mit immer schwerer zu bestimmenden Wahrscheinlichkeiten wechselseitig überlagern werden. Kein technischer Fortschritt ist bekannt, der ein solches Szenario von »Schrödingers Schweigepflicht«[91] zwangsläufig auslöst. Kerckhoffs' Prinzip und die Gesetze der Mathematik stehen dem entgegen, soweit man sie anwendet.

91 https://de.wikipedia.org/wiki/Schr%C3%B6dingers_Katze

# 5. Dokumentation für Programmierer

## 5.1. Model-View-ViewModel

OpenDear folgt dem Software-Entwurfsmuster Model-View-ViewModel (MVVM)[92]. Das spiegelt sich im Aufbau der Projektmappe wider, die die Unterordner »Model«, »View« und »ViewModel« enthält. Zusätzlich liegen in einem Ordner »Assets« die vom View benötigten Bilder und Symboldateien.

Anstelle des Quellkodes der Klassen »RelayCommand« und »ViewModelBase« zur Realisierung des MVVM-Entwurfsmusters hätte das NuGet-Paket »MvvmLightLibs« Verwendung finden können. Auf diese zusätzliche Abhängigkeit habe ich jedoch verzichtet.

Unter »Model« versteht man die eigentliche Programmlogik, die so weit wie möglich unabhängig vom konkreten Benutzer sein soll. In diesem Projekt gehören zum Model unter anderem zu vergleichende Paare von Dateien, kryptographische Schlüssel und das Ansteuern der Kryptographieschnittstelle des Betriebssystem. Nicht zum Model gehört, ob der Nutzer den Typ eines kryptographischen Schlüssels auf Deutsch als »symmetrisch« oder auf Englisch als »symmetric« angezeigt bekommt, ob er mit einer Maus auf eine graphische Benutzeroberfläche klickt oder am anderen Ende eines Netzwerks vor einer Tastatur sitzt. Das Model ist in der Programmiersprache C# geschrieben.

Im Gegensatz dazu ist »View« die graphische Benutzeroberfläche mit ihren Bildern, Textfeldern und Schaltflächen. Der View soll so weit wie möglich passiv, ohne eigene Programmlogik sein und ist in der Beschreibungssprache Extensible Application Markup Language (XAML) geschrieben.

Zwischen »View« und »Model« vermittelt das »ViewModel«, das Daten aus der Programmlogik an die Benutzeroberfläche liefert, aber weder die Abläufe im Model noch den Aufbau des View kennen soll. Außerdem prüft und verknüpft das ViewModel Eingabedaten und schreibt sie zurück in das Model. Es ist in der Programmiersprache C# geschrieben.

## 5.2. Namenskonventionen

Bislang bin ich nicht davon abzubringen, dass ich mit meiner Groß- bzw. Kleinschreibung für Bezeichner von den Microsoft-Richtlinien[93] abweiche. Die zugehörige Warnung »IDE1006« habe ich deshalb in den Projekteigenschaften unterdrückt. Unverändert gilt, dass Namen von Methoden (= Teilprogrammen ohne Rückgabewert), Funktionen (= Teilprogrammen mit Rückgabewert) und Klassen (= Abbildung realer Objekttypen in Software) mit einem Großbuchstaben beginnen.

Variablen stelle ich ihren Datentyp in Form eines Kürzels aus einem oder zwei Kleinbuchstaben voran. Das hat den Vorteil, dass ich für gleiche Sachverhalte nicht verschiedene Variablennamen erfinden muss, sondern z.B. *iLengthInBits* schreiben kann für die interne

92 https://de.wikipedia.org/wiki/Model_View_ViewModel

93 https://docs.microsoft.com/de-de/dotnet/standard/design-guidelines/capitalization-conventions

Zahlendarstellung und *sLengthInBits* für die Darstellung derselben Information als Zeichenkette auf der Benutzeroberfläche. *ltKeys* bezeichnet die programminterne Liste aller Schlüssel, *blKeys* die an die Benutzeroberfläche gebundene *BindingList* aller zum gewählten Verschlüsselungsverfahren passenden Schlüssel. Kann ich mich nicht an den Namen eines von mir vergebenen Bezeichners erinnern, ist dessen Datentyp dennoch meist aus dem Kontext ersichtlich, so dass der entsprechende Anfangsbuchstabe meine Suche eingrenzt. Zudem erleichtert diese Schreibweise das Verständnis des Programmkode außerhalb einer Entwicklugsumgebung, die den Datentyp einer Variablen automatisch als Tooltip anzeigt. Die von mir gewählten Kürzel habe ich im Folgenden zusammengestellt.

| **Datentyp** | **Typname** | **Kürzel** |
|---|---|---|
| Konstante eines unten aufgeführten Typs, z.B. double | const double | cdBezeichner |
| *hardwarenah* | | |
| Wahrheitswert | bool | isBezeichner |
| Byte | byte | bBezeichner |
| *Zahlen* | | |
| selbst definierte Aufzählung von Zuständen | enum nBezeichner | eBezeichner |
| 16-Bit-Ganzzahl mit Vorzeichen | short | hBezeichner |
| 32-Bit-Ganzzahl mit Vorzeichen | int | iBezeichner |
| 64-Bit-Ganzzahl mit Vorzeichen | long | kBezeichner |
| 16-Bit-Ganzzahl ohne Vorzeichen | ushort | tBezeichner |
| 32-Bit-Ganzzahl ohne Vorzeichen | uint | uBezeichner |
| 64-Bit-Ganzzahl ohne Vorzeichen | ulong | vBezeichner |
| 32-Bit-Fließkommazahl | float | fBezeichner |
| 64-Bit-Fließkommazahl | double | dBezeichner |

| Datentyp | Typname | Kürzel |
|---|---|---|
| *Text* | | |
| einzelnes Zeichen | char | cBezeichner |
| Zeichenkette | string | sBezeichner |
| *komplexe Objekte* | | |
| Array = »Feld« eines oben aufgeführten Typs, z.B. double | double[] | adBezeichner |
| Befehl | RelayCommand | rcBezeichner |
| Ergebnis einer LINQ-Abfrage | IEnumerable<> | qyBezeichner |
| Fehlerzustand | Exception | exBezeichner |
| Liste | List<> | ltBezeichner |
| an die Benutzeroberfläche gebundene Liste | BindingList<> | blBezeichner |
| Warteschlange | Queue<> | quBezeichner |
| Wörterbuch | Dictionary<> | dyBezeichner |

### 5.3. Abhängigkeiten

Die von OpenDear vorausgesetzten NuGet-Pakete können in Visual Studio mit dem Menüpunkt »Projekt => NuGet-Pakete verwalten…« eingesehen werden. Zum einen benötigt die zugrundeliegende SQLite-Datenbank das Paket »System.Data.SQLite.Core«. Zum anderen setzt die Arbeit mit OpenPGP-SmartCards das Paket »Pkcs11Interop« voraus.

Sollen OpenPGP-SmartCards wie der NitroKey Verwendung finden, muss die OpenSC-Bibliothek[94] auf dem Rechner installiert sein.

Die benötigten Funktionen des MVVM-Entwurfsmusters habe ich direkt als Programmkode eingefügt und auf das Paket »MvvmLightLibs« verzichtet.

94 https://github.com/OpenSC/OpenSC/wiki