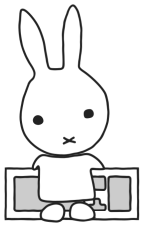
Repression

Angewandte Kryptographie

Eine Einführung in und Anleitung zur effektiven Computersicherheit und Repressionsschutz

Alan Schwarz



compassion media

1. Auflage, April 2021

Text, Satz & Illustrationen: Alan Schwarz
Verwendete Emojis: OpenMoji.org (CC BY-SA 4.0)
Korrektorat: Magazin TIERBEFREIUNG
Druck & Distribution: compassion media, Münster

Digitale Version und LaTeX Quellen:
`https://0xacab.org/alan_schwarz/angewandte_kryptographie`

Dieser Text erschien ursprünglich im Magazin TIERBEFREIUNG 109.

Inhaltsverzeichnis

1 Einleitung

Ein Blick in die beinahe prähistorisch anmutende Vergangenheit von 2013 lohnt sich selbst heute noch. Ein Fundus hoch geheimer Dokumente wird vom damaligen CIA-Subunternehmer Edward Snowden an die Öffentlichkeit gebracht. Diese enthalten größtenteils Interna der US-amerikanischen Sicherheitsbehörde NSA. Dabei sind für unsere Zwecke zwei Folien von besonderer Bedeutung. Abbildung 1 zeigt die Möglichkeiten systematischen Abgreifens beliebiger Informationen im Rahmen des PRISM-Programms. Interessant ist hierbei, dass dabei keine komplizierten und unverständlichen Methoden verwendet werden. Unter dem bis heute geltenden Patriot Act kann die NSA schlicht in den USA sitzende Unternehmen zwingen alle Daten preiszugeben, die die NSA einfordert. Ob Microsoft, Google, Apple oder Facebook, der Zugang zu Daten *muss* gewährt werden. Dadurch können NSA und mit ihr kooperierende Geheimdienste wie etwa der britische GCHQ und der deutsche BND an private Daten von Benutzer*innen gelangen, selbst wenn Unternehmen versprechen diese sicher aufzubewahren.

Gleichzeitig gibt uns ein Vortrag der NSA vom Juni 2012 Einblick in die Grenzen der Abhörung einer der größten und bestbezahlten Abhörorganisationen der Welt. Abbildung 2 ist ein Auszug dieser Präsentation, welche grafisch anschaulich zeigt, vor welchen Problemen der Dienst 2012 stand. Auch wenn diese Angaben acht Jahre später durchaus mit Vorsicht zu genießen sind, scheint die Kombination verschiedener Systeme (auf der ganz rechten Spalte) zu einem aus Abhörsicht „katastrophalen“ Ergebnis zu führen.

Im Folgenden werden wir uns mit einigen hiervon beschäftigen und dabei einen Einblick in ihre Funktionsweise sowie eine kurze Anleitung zu ihrer Verwendung geben. Es sollte jedoch beachtet werden, dass Abbildung 2 hier lediglich motivierenden Charakter hat und keinesfalls mit einer aktuellen Sicherheitsempfehlung verwechselt werden darf. Einige der dort genannten Schlagwörter (etwa TrueCrypt) sind veraltet bzw. von Alternativen abgelöst worden und

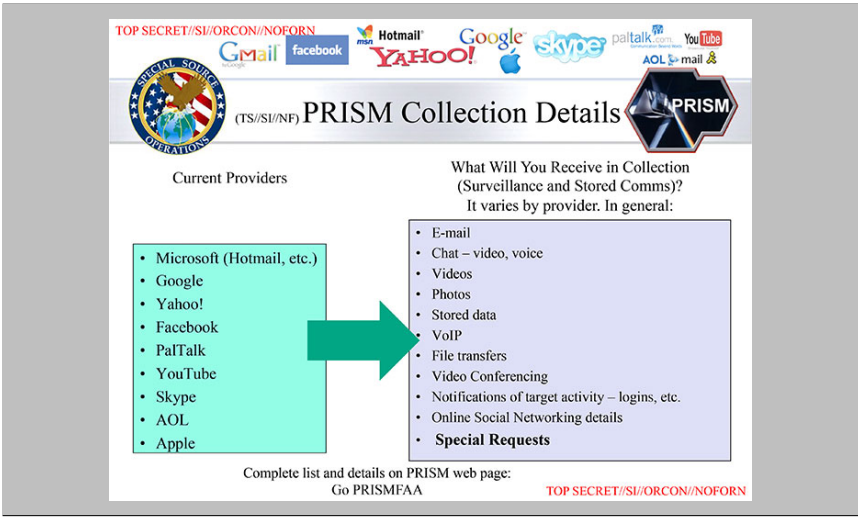


Abbildung 1: PRISM. Quelle: Edward Snowden, Washington Post [1]

wiederum andere (etwa Mail.ru) sind aus gänzlich anderen Gründen kein hilfreiches Werkzeug bzw. stehen stellvertretend für Dienste, die nicht unter der Kontrolle der NSA stehen.

Zum derzeitigen technologischen Stand sind die meisten dieser Systeme nur wirklich sicher benutzbar, wenn wir die Grundzüge der darunterliegenden Funktionsweise verstehen, da sonst kritische Fehler in ihrer Benutzung gemacht werden können. Daher muss dieser Text etwas mehr Grundlagen schaffen als für die bloße Anwendung zunächst erforderlich scheint. Dafür wechseln sich Theorie und Praxis im Folgenden immer wieder ab; Abschnitte sind entsprechend mit einem T oder P markiert.

Computersicherheit geht uns alle an: es ist nicht nur Selbstschutz, sondern auch Schutz unserer Freund*innen, Genoss*innen, Mitstreiter*innen und Kompliz*innen.

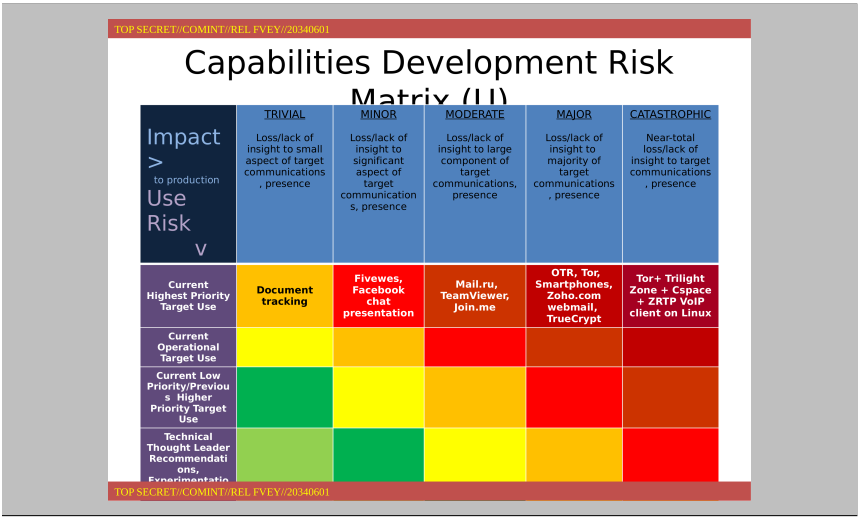


Abbildung 2: Risikoanalyse NSA. Quelle: Edward Snowden [2]

2 Systemeinrichtung

2.1 Freie und Offene Software (T)

Egal, wie sicher und ausgeklügelt die Verfahren auch sein mögen, die wir einsetzen wollen: Maßgeblich dafür, wie gut diese in der Praxis sind, hängt von ihrer Implementation ab. Eine (Software-) Implementation besteht dabei konzeptionell und sehr grob aus drei Schritten:

1. die Formulierung einer Idee,
2. die Übersetzung dieser Idee in eine Sprache, die sowohl Menschen wie auch Computer lesen können (diese wird als Quelltext bezeichnet) und
3. die Übersetzung dieser Formulierung in eine effektiv nur noch von Computern verständliche Sprache.

Schritt 3 kann mit relativ wenig Aufwand vollständig automatisch von Computern erledigt werden – dieser Schritt wird als Kompilation bezeichnet. Die Kompilation umzukehren, also wieder etwas Menschenlesbares zu gewinnen, ist in der Praxis jedoch aus verschiedenen Gründen quasi unmöglich. Wir können uns dies am Beispiel eines Webbrowsers verdeutlichen; zum Beispiel Firefox. Wenn ich mit Firefox die Webseite `tierbefreiung.de` betrachten möchte, muss ich eine Datei auf meinem Computer haben, welche das Firefox-Programm in seiner kompilierten Form enthält. Diese Version kann ein Mensch jedoch weder sinnvoll lesen noch sinnvoll verändern. Wenn ich also das Verhalten des Browsers ändern oder auch nur überprüfen möchte, muss ich irgendwie an den Quelltext kommen.

Hierin liegt das Konzept von *Free and Open Source Software* (FOSS) begründet. Freie und offene Software unterscheidet sich von *Proprietärer Software* (unfreie bzw. geschlossene Software) dadurch, dass der Quelltext frei verfügbar ist und weiter, dass jede Person eine veränderte Kopie hiervon anbieten kann, solange sie stets den Quelltext uneingeschränkt verfügbar macht (mehr Details: [3, 4]). In der Praxis bedeutet dies, dass unabhängige Menschen den Quelltext auf Probleme überprüfen können, und dass diese auch zugleich die Probleme reparieren können. Solche Probleme können unbeabsichtigt sein (dann nennen wir sie Fehler) und sie können beabsichtigt sein (dann nennen wir sie Hintertüren).

Diese Art der öffentlichen Prüfung ist bei proprietärer Software quasi unmöglich. Wer also unfreie Software von beispielsweise Microsoft oder Apple installiert, muss gänzlich auf die Expertise der Mitarbeitenden dieser Konzerne vertrauen und zusätzlich darauf vertrauen, dass diese Konzerne nicht absichtlich Hintertüren und andere Abhörmaßnahmen eingebaut haben. Im Gegensatz hierzu wird FOSS von Gemeinschaften gepflegt, die das Beisteuern von Sicherheits- und Funktionsverbesserungen („Patches“) ermöglichen, aber kuratieren und somit das böswillige Unterjubeln von Schadcode verhindern. FOSS darf übrigens nicht mit sogenannter

„Freeware“ verwechselt werden, da bei dieser zwar das Kompilat kostenlos verfügbar ist, der Quelltext jedoch nicht.

Ein sicheres System mithilfe von unfreier Software aufzusetzen ist von Anfang an zum Scheitern verurteilt. Insbesondere, wenn bereits das Betriebssystem proprietär ist, sind die meisten, wenn nicht alle, in diesem Text folgenden Anleitungen tendenziell nutzlos. Hinzu kommt, dass häufig Geschlossenheit und Kommerzialisierung von Software einhergehen, sodass weitere Software häufig ebenfalls der kommerziellen Verwertungslogik unterliegt. Im Kontrast hierzu ist insbesondere die *Free Software* Bewegung von Anfang an eine politische, welche sich nicht ausschließlich für die praktischen Vorteile von FOSS interessiert, sondern auch für die gesellschaftlichen und ethischen Implikationen. Für eine detailliertere Positionierung wird auf die Ausführungen der *Free Software Foundation* (FSF) verwiesen. [4] Ungeachtet der zum Teil wichtigen (philosophischen) Unterschiede zwischen „Free Software“ bzw. „Open Software“ werden diese jedoch häufig synonym zueinander, zum Oberbegriff FOSS sowie der neueren Bezeichnung als „Sofware Libre“ verwendet.

2.2 Betriebssysteme (P)

Es gibt eine Reihe von FOSS-Betriebssystemen. Das hier relevanteste dürfte *GNU/Linux* sein (häufig fälschlich auch nur als „Linux“ bezeichnet). Anders als bei kommerziellen Systemen, wird GNU/Linux von verschiedenen Gruppen / Initiativen / Stiftungen in verschiedenen Konfigurationen / Ausfertigungen / Geschmacksrichtungen angeboten – diese werden als *Distributionen* bezeichnet. Aufgrund der freien bzw. offenen Natur von GNU/Linux-Distributionen können diese gegenseitig voneinander profitieren – Patches verschiedener Distributionen können frei im „Linux-Ökosystem“ in andere einfließen. Von der Vielzahl an existierenden Distributionen wollen wir uns an dieser Stelle nur zwei relevante herausgreifen: Debian und Tails. Alternativen hierzu sind etwa Ubuntu, Mint, Arch-Linux, Fedora, Gentoo oder SUSE.

Da Tails eine sehr spezielle Aufgabe erfüllt und ein Verständnis nachfolgender Kapitel erfordert, werden wir es erst zum Schluss dieser Broschüre wieder aufgreifen. Debian hingegen ist eine der stabilsten und weitverbreitetsten Distributionen, die auch für den alltäglichen und komfortablen Gebrauch geeignet ist. Daher empfehle ich Debian anstatt der verbreiteten „Einstiegsdistribution" Ubuntu. Da Ubuntu im Kern aber auf Debian beruht, sind viele Internetanleitungen für Ubuntu, von denen es gerade in deutscher Sprache sehr viele gibt, häufig auf Debian übertragbar. Ein Installationsmedium kann frei von der offiziellen Webseite [5] heruntergeladen werden[1].

Die Installationswerkzeuge von Debian und anderer moderner Distributionen sind auch für Laien mit etwas Einarbeitung in einem Nachmittag umsetzbar – insbesondere unter Zuhilfenahme einer der zahllosen Internetanleitungen (etwa [6] und [7]). Auf jeden Fall sollte aber vor einer Installation eines neuen Systems vorsichtshalber eine Datensicherung auf ein externes Medium durchgeführt werden.

Unter anderem praktisch für den Umstieg auf GNU/Linux von proprietären Systemen wie Windows ist, dass Debian einen sogenannten *Bootloader* namens GNU GRUB 2 mitbringt, welcher beim Hochfahren des Computers dem Start des Betriebssystems vorgeschaltet wird. Dieser erlaubt es, mehrere Betriebssysteme (zum Beispiel Debian und Windows) auf dem selben Computer installiert zu haben und für den Start des Computers das gewünschte System auszuwählen. So lässt sich durch Neustarten beliebig zwischen einem alten Betriebssystem und dem neu installierten wechseln. Diese wohnen auf unterschiedenen Partitionen (siehe [7]) der SSD oder Festplatte und laufen unabhängig voneinander. Dies wird als *Dual-Boot* bezeichnet.

[1] Hinweis: Es sollte auf jeden Fall nicht die Version Debian 10/buster heruntergeladen werden, sondern mindestens die Nachfolgeversion Debian bullseye. Siehe `https://www.debian.org/devel/debian-installer/index.de.html`

Viele Distributionen (u. a. Debian, Ubuntu, Tails) ermöglichen auch das Starten eines Betriebssystems ohne jedwede Installation. Dies wird als *„live"-System* bezeichnet. GNU/Linux startet dann direkt von etwa einem USB-Stick und hinterlässt nach dem Herunterfahren keinerlei Spuren auf dem Computer. Dies ermöglicht u.;a. auch verschiedene Distributionen und Oberflächen „unverbindlich" einmal auszuprobieren, ohne Änderungen am aktuell installierten System vornehmen zu müssen. Mehr hierzu unter Tails (Abschnitt 5.1). Ein „live"-System ist jedoch meist etwas langsamer als ein System, welches dauerhaft auf einer Festplatte installiert ist.

Unabhängig von der gewählten Distribution lässt sich bei der Installation und später bei der Anmeldung aus einer Fülle verschiedener Desktopumgebungen wählen, die primär verschiedenen Geschmäckern und Präferenzen dienen. Debian unterstützt derzeit sieben verschiedene Desktopumgebungen – wer sich von dieser Auswahl überwältigt fühlt, möge MATE oder das extrem leichtgewichtige LXDE ausprobieren. Egal welche Distribution oder Oberfläche gewählt wird, sollte das System stets auf dem aktuellsten Stand gehalten werden, um Sicherheitsupdates und neue Softwareversionen zu erhalten. Unter Debian lässt sich mit der Paketverwaltung Synaptic (unter Systemverwaltung) mit wenigen Klicks das System aktualisieren. Um alle Programme, die in diesem Text empfohlen werden, installieren zu können sollten dort die Paketquellen von „main" auf „main contrib" ergänzt werden.

2.3 Grundlagen Symmetrischer Kryptographie (T)

Nachdem wir ein sicheres Betriebssystem installiert haben, wollen wir vielleicht hiermit über das Internet kommunizieren und dafür sorgen, dass außer uns niemand auf unser System zugreifen kann. Bevor wir die praktische Umsetzung dieser Vorhaben untersuchen, müssen wir zunächst etwas Grundlagen schaffen. Denn fast alle nachfolgenden Abschnitte werden sich in irgendeiner Form auf *Kryptographie* beziehen.

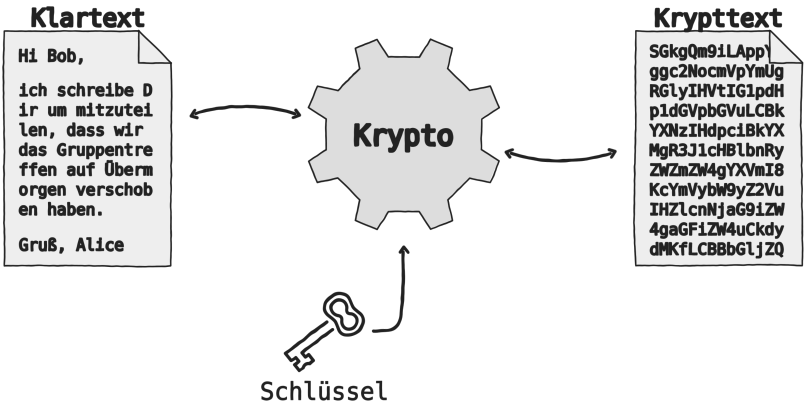


Abbildung 3: Grundkonzept moderner Kryptographie

Ein kryptographisches System dient dazu, Daten vor Unbefugten zu schützen. Zunächst ist Kryptographie von sogenannter *Steganographie* abzugrenzen: Letztere versucht nicht nur die tatsächlichen Daten vor unbefugtem Zugriff zu schützen, sondern auch deren bloße Existenz zu verbergen. Zum Beispiel könnte in diesem Text durch subtile Interpunktionsfehler eine geheime Nachricht kodiert sein, die nur eigeweihte Lesende entdecken können. Alle anderen würden nicht einmal merken, dass eine geheime Nachricht überhaupt existiert.

Im Gegensatz hierzu hat Kryptographie nicht das Ziel, das Vorhandensein von Daten zu verschleiern. Im Gegenzug können daher beliebig aggressive Verschlüsselungsverfahren eingesetzt werden, die vor unerlaubter Manipulation oder unerlaubtem Zugriff auf den *Klartext* schützen. Ein auf diese Art geschützter Klartext wird als *Krypttext* oder *verschlüsselter Text* bezeichnet — siehe Abbildung 3. Um einen Klartext in einen Krypttext zu verwandeln (*Verschlüsselung*) oder umgekehrt (*Entschlüsselung*) erfordert jedes Verfahren die Kenntnis eines Geheimnisses, genannt *Schlüssel* (engl: *key*). Häufig sind diese Schlüssel lediglich große zufäl-

lig gewählte Zahlen — abhängig vom Verfahren sind dies Zahlen mit hundert oder tausenden Stellen. Ohne Kenntnis des richtigen Schlüssels lässt sich der Krypttext nicht entschlüsseln und der Klartext nicht verschlüsseln.

Prinzipiell werden kryptographische Verfahren in *symmetrisch* und *asymmetrisch* unterteilt. Symmetrische Verfahren verwenden zur Ver- und Entschlüsselung den genau gleichen Schlüssel. Asymmetrische Verfahren verwenden unterschiedliche Schlüssel, welche die Eigenschaft haben, dass sich aus der Kenntnis eines der beiden Schlüssel *nicht* der jeweils andere ableiten lässt. Mehr dazu in Abschnitt 4.1.

Die Funktionsweise von symmetrischen Verfahren beruht darauf, dass ein zuvor bekannter Schlüssel auf einem sicheren Weg ausgetauscht wird. Ein intuitives Beispiel eines solchen sicheren Wegs könnte etwa sein, den Schlüssel sicher auf einem USB-Stick zu verstauen und einer anderen Person in die Hand zu drücken, welche den Schlüssel nach dem Kopieren auf den eigenen Computer wieder vom USB-Stick löscht. In der Praxis ist ein solcher Austausch zu aufwendig, sodass symmetrische Verfahren selten alleine auftreten. Ist der Schlüssel erst einmal ausgetauscht, können im Nachhinein Nachrichten mit diesem verschlüsselt werden und über ungesicherte Kanäle geschickt werden (Abbildung 4). Im Folgenden gehen wir davon aus, dass eine Person namens *Alice* eine solche geheime Nachricht an eine Person namens *Bob* schicken möchte. Eine Person namens *Eve* möchte diese Nachricht abhören.

Zur Veranschaulichung symmetrischer Verfahren betrachten wir One-Time-Pad (OTP). OTP ist eines der ältesten und simpelsten Verfahren der Geschichte und zugleich das buchstäblich einzige Verfahren weltweit, welches beweisbar nicht brechbar ist. Wenn es so großartig ist, wieso wird es dann nicht ausschließlich eingesetzt? Der Schlüssel muss die gleiche Länge wie der Klartext haben, darf nur ein einziges Mal verwendet werden und muss danach weggeworfen werden. Für jeden Buchstaben im Klartext enthält der

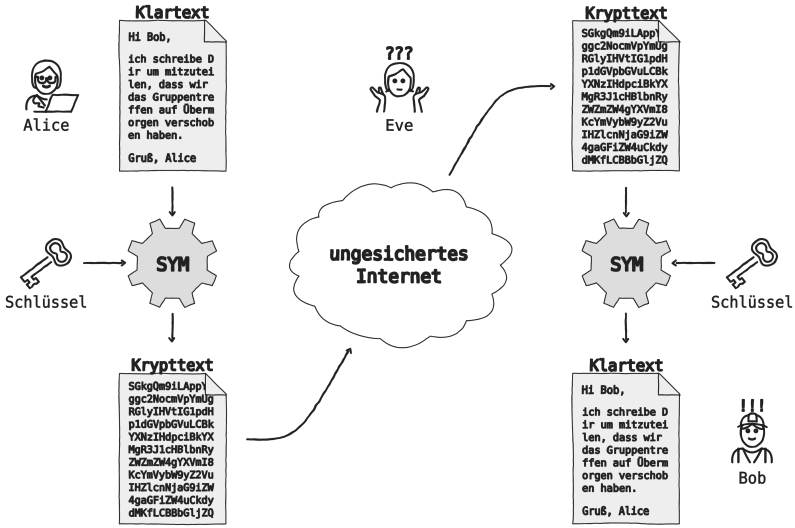


Abbildung 4: Aufbau symmetrischer Kryptographie

Schlüssel eine Zahl, die festlegt, um wieviel dieser eine Buchstabe verändert wird. Zum Beispiel, wenn im Klartext ein „c“ steht und im Schlüssel eine 4, dann wird das „c“ um vier Buchstaben im Alphabet verschoben, also durch ein „g“ ersetzt. Ein „y“ im Klartext und eine 7 im Schlüssel würde den Buchstaben „f“ erzeugen, und so weiter[2]. Das Problem: Würde der Schlüssel mehrfach benutzt, ließe sich sowohl der Klartext wie auch der Schlüssel mit vergleichsweise einfacher statistischer Analyse aus mehreren Krypttexten ableiten.

In der Praxis ist OTP also alleine nicht hilfreich. Das Prinzip wird jedoch bis heute von modernen, sogenannten *Stream-Chiffren* wie etwa dem populären und FOSS-implementierten ChaCha20 eingesetzt. Ein Stream-Chiffre ist ein symmetrisches Verschlüsselungsverfahren, welches aus einem vergleichsweise kleinen Schlüssel (256 bit bzw. eine Dezimalzahl mit 78 Stellen) und einer kleinen

[2] Hinweis: In der Realität wird anstatt einer Addition von Klartext und Schlüssel eine bitweise *Verxoderung* durchgeführt, doch das Prinzip ist dasselbe.

für jede Übertragung neu gewählten zufälligen Zahl (genannt *Nonce*) einen unendlichen OTP-Schlüssel erzeugt, der von einem echt zufälligen Schlüssel nicht von außen unterscheidbar ist. Weiß das Gegenüber den ursprünglichen Schlüssel und das Nonce (welches im Klartext übertragen werden darf), kann es den gleichen unendlichen OTP-Schlüssel ableiten und damit aus dem Krypttext wiederum den Klartext entschlüsseln.

Wichtige symmetrische Krypto-Verfahren sind neben ChaCha20 bis heute AES, Blowfish / Twofish und IDEA[3]. Unabhängig von der detaillierten Funktionsweise verschiedener Chiffren ist modernen symmetrischen Verfahren gemein, dass sie den Klartext auf möglichst zufällig aussehende Weise verändern, durchschütteln und -rühren, so dass ohne Kenntnis des Schlüssels vom Krypttext kein Rückschluss auf den Klartext möglich ist.

2.4 Festplattenverschlüsselung (P)

Die Tierbefreiungsbewegung eint, dass wir nicht oder nur selten die Symptome von Ausbeutung direkt bekämpfen, sondern versuchen, unmittelbar an die Wurzel zu greifen. Eine solch *radikale* Herangehensweise hat den Vorteil, sich nicht vor der Sisyphusarbeit unzähliger Manifestationen zu verzetteln. Im selben Sinne radikal können auch Angriffe auf Sicherheitssysteme verlaufen. Wer die Quelle von Nachrichten kontrolliert, braucht sich keine Gedanken machen, wie ein Verschlüsselungsverfahren zu brechen ist und hat langfristiger etwas davon.

Unzählige Male wurden Sicherheitssysteme ausgehebelt, indem entweder das Betriebssystem angegriffen wird oder indem Repressionsorgane Menschen zuhause besuchen und Hardware unter Androhung von Waffengewalt rauben (sie nennen das dann „konfiszieren“). Wer eine aktuelle GNU/Linux-Distribution betreibt, hat schon einmal eine deutlich bessere Chance, dass in den PC nicht von außen eingestiegen wird. Vor physischem Zugriff auf den PC kann

[3] Hierbei handelt es sich jedoch um *Block-Chiffren*, nicht um Stream-Chiffren

dies allerdings nicht schützen. Auch wenn sich das System nicht automatisch nach dem Start anmeldet (was auf jeden Fall und ohne jede Diskussion so sein sollte) kann aus einem entwendeten Computer die Festplatte problemlos ausgelesen werden – selbst ohne den Computer aufzuschrauben, mithilfe eines live-Systems (siehe oben). Schlimmer noch, es kann Schadsoftware installiert werden und anschließend der Computer zurückgegeben werden. Wenn also eine zwielichtige Partei ein Gerät in die Finger bekommt, führt *nichts* an einer vollständigen Neuinstallation vorbei. Empfehlenswert wäre sogar die Hardware von Expert*innen genauestens untersuchen zu lassen oder gleich durch neue Hardware zu ersetzen.

Um bestehende Daten vor etwa unangemeldeten Hausbesuchen zu schützen, sollte die Festplatte, zumindest die Datenpartition, verschlüsselt werden. In Abbildung 2 finden wir Festplattenverschlüsselung durch das inzwischen nicht mehr fortgeführte (und daher nicht zu empfehlende) TrueCrypt repräsentiert.

Unter GNU/Linux hat jede*r Benutzer*in ein eigenes „home"-Verzeichnis. Zum Beispiel, wenn Alice und Bob Konten auf dem gleichen System haben, fänden sie ihre persönlichen Dokumente unter dem Pfad `/home/alice` bzw. `/home/bob`. Zur Verschlüsselung von Alices Daten verschlüsseln wir daher nur die Dateien im Ordner `/home/alice`. Dieses Verzeichnis zu verschlüsseln reicht meistens, da unter GNU/Linux persönliche Daten und persönliche Konfigurationen (inklusive Mails, Browserhistorie und Dokumente) stets unter diesem Pfad abgelegt werden.

Dies leistet (unter Debian und Ubuntu) das Programm `ecryptfs`: Einmal installiert ist im Alltag kein Unterschied zu einer unverschlüsselten Festplatte zu merken, denn der Umstand, dass die Daten verschlüsselt gespeichert sind, ist in der Benutzung nicht sichtbar. Dies wird als *transparente Verschlüsselung* bezeichnet. Nachdem Alice sich mit einem (sicheren!) Passwort anmeldet, sind alle Dateien unter diesem Pfad sichtbar — und sonst nicht.

Um `ecryptfs` einzurichten, müssen zunächst die Pakete `rsync` und

`ecryptfs-utils` aus den Paketquellen installiert werden[4]. Dies geht unter Debian am einfachsten mit der Paketverwaltung „Synaptic“. Für die simpelste Variante zur Einrichtung, muss Alice sich zunächst abmelden, als der Adminuser „root“ einloggen und den Befehl

```
ecryptfs-migrate-private -u alice
```

einmalig ausführen – dies wird mit `ecryptfs` direkt mitinstalliert. Es richtet automatisch sinnvolle Parameter ein, zum Beispiel wählt es als Verschlüsselungsverfahren AES128. Nach der nächsten Anmeldung werden sämtliche Dateien unter `/home/alice` automatisch und transparent verschlüsselt. Wichtig ist, dass Alice sich vor dem nächsten Neustart des Computers anmeldet, um die Datenmigration abzuschließen.

3 Das Web

3.1 Der grundlegende Aufbau des World Wide Webs (T)

Mit dem Aufkommen des Konzeptes vom Web 2.0 und Social Media wurde der vergleichsweise simple Aufbau des Webs, mit seinen statischen HTML-Seiten, für einen Zweck eingesetzt, für den er ursprünglich nicht konzipiert war. Das bedeutet, dass einiges an Funktionalität im Nachhinein angetackert ist. Zusätzlich hat sich die Technologie, wie Webseiten an sich aufgebaut sind, massiv weiterentwickelt — wer sich an die ersten Webseiten aus den 90ern erinnert, kann dies bestätigen.

Um eine Webseite anzuzeigen, muss (mindestens) eine Datei von (mindestens) einem Server auf den eigenen Computer übertragen werden. Dies bewerkstelligt das vollständig unverschlüsselte Transportprotokoll HTTP. Nachdem die Datei übertragen ist, ist die Transaktion aus technischer Sicht allerdings abgeschlossen: Es wird (aus HTTP-Sicht) nicht vorgehalten, wer welche Seite

[4] `ecryptfs-utils` ist nicht in Debian 10/buster verfügbar, sondern erst wieder mit der neueren „bullseye“ Version.

aufgerufen hat. Dann analysiert der Browser (z.B. Firefox, Chrome/Chromium, Safari, etc.) die Datei und beginnt sie darzustellen. Dazu ist es praktisch immer notwendig zusätzliche Dateien von potentiell anderen Servern herunterzuladen: zum Beispiel Bilder, Stylesheets (Anleitungen dafür, wie der Inhalt dargestellt werden soll), Skripte, etc. Dazu baut der Browser jeweils neue Verbindungen auf und lädt die neuen Dateien herunter. Bei interaktiven Seiten, was heutzutage alle Social-Media-Seiten sind, steht der Browser darüber hinaus permanent mit einem oder mehreren Servern in Verbindung; er übermittelt Daten und fordert dynamisch neue Inhalte an.

Dieser Aufbau sieht zunächst aber nicht vor, dass Inhalte für verschiedene Benutzer*innen unterschiedlich aussehen oder dass verschiedene Benutzer*innen verschiedenen Inhalt sehen. Ein Einloggen kann kaum sinnvoll abgebildet werden. Die Lösung hierfür stellen Cookies und Sessions dar: Hiermit kann konzeptionell eine Sitzung ermöglicht werden, die sich über mehrere Seitenabrufe hinweg aufrechterhalten lässt. Ein Cookie ist eine Identifikationsmöglichkeit (z.B. eine Zahl oder eine kurze Zeichenkette), die vom Server an den Browser ausgestellt wird und die dieser bei jeder erneuten Anfrage an den Server mitschickt. Damit kann sich etwa Alice bei einem Server einloggen und erhält (falls das Kennwort stimmt) ein Cookie geschickt, welches der Browser intern speichert. Bei der nächsten Anfrage erkennt der Server das mitgeschickte Cookie und weiß damit, dass die Anfrage erneut von Alice kommt und kann somit den passenden Inhalt ausliefern.

Zusätzlich zu Browser-Cookies gibt es noch eine Vielzahl anderer Methoden, mit denen eine Person identifiziert werden kann. Informationen lassen sich auch in die URL einbetten, wenn ein Objekt angefordert wird. Zum Beispiel ein eingebettetes Bild, welches unter `example.com/bild.jpg` liegt, kann stattdessen auch angefordert werden als: `example.com/bild.jpg?id=65723534377352`. Der Server weiß dann, dass `bild.jpg` angefordert wurde und liefert dies aus. Der Rest der URL kann verwendet werden, um wei-

tere Daten zu übertragen; zum Beispiel eine Identifikationszahl 65723534377352. Hiermit können auch verschiedene Server miteinander über den Umweg eines Browsers kommunizieren. Eine Webseite `a.de` könnte etwa ein Bild `b.de/b.jpg?user=alice` einbinden und dem Server unter `b.de` erlauben dessen eigene bei Alice gesetzten Cookies mit der Identität von Alice zu verknüpfen.

Diese Techniken können von sogenannten *Aggregatoren/Trackern* verwendet werden, um Informationen über eine Person zusammenzutragen. Indem ein Dienst von einer Seite angeboten wird, der für viele Seitenbetreibende nützlich wirkt, wie etwa Google Analytics, Facebook-Like-Button, etc. kann eine extrem hohe Abdeckung des Webs erzielt werden. Es kann aktuell davon ausgegangen werden, dass Unternehmen wie Google, Akamai oder Facebook Abdeckungen von nahezu 100% haben[5]. Das bedeutet, dass Webseiten in irgendeiner Form Inhalt von einem Aggregator wie z.B. Google einbinden. Sei es ein kostenlos angebotenes Skript, welches die Erstellung der Webseite einfacher macht, seien es Grafiken oder Stylesheets. Wenn Google ein Cookie in Alices Browser platziert, wird der Google-Server dadurch detailliert über das Onlineverhalten von Alice informiert, erstellt ein Profil und kann den eigenen Kund*innen „bessere" auf Alice zugeschnittene Werbeanzeigen anbieten, wodurch diese ihre Werbeeinnahmen erhöhen können.

Zusätzlich dazu kann ein Aggregator hierdurch Personen *deanonymisieren*. Wenn Alice auf 99 Seiten anonym ist und dann auf einer Seite, etwa einem Online-Shop, persönliche Informationen hinterlässt, könnte ein solcher Aggregator aus den gesammelten Daten Alices Identität in Verbindung zu allen 99 anderen Seitenbesuchen setzen.

[5] Eine nennenswerte Ausnahme ist Wikipedia und ihre assoziierten Schwesterprojekte, welche aus gutem Grund spendenfinanziert sind. Stichproben ergeben auch, dass viele Bewegungsshops wie *rootsofcompassion.org*, *tierbefreiershop.de* oder *schwarzesocke.org* erfreulich sauber zu sein scheinen.

3.2 Eine mäßig sichere Verwendung des Webs (P)

Sobald wir einen verhältnismäßig sicheren Computer mit einem aktuellen GNU/Linux und einer verschlüsselten Festplatte haben, können wir uns überlegen, wie das Web sicher zu verwenden ist — zunächst erst einmal *mäßig* sicher, weiter unten steigern wir uns zu *ziemlich*. Die Grundvoraussetzung ist es, einen sicheren und freien Browser zu installieren, der zusätzlich erweiterbar ist. Effektiv bedeutet dies aktuell: Firefox. Dieser ist bei Debian und den meisten anderen Distributionen bereits vorinstalliert.

Als wichtigste Maßnahme, um Datenverkehr zu schützen, sollten Webseiten stets per HTTPS abgerufen werden. Im Gegensatz zu dem vollkommen unverschlüsselten HTTP wird die gesamte Kommunikation bei HTTPS über das TLS Protokoll (früher SSL) verschlüsselt. Beim Aufruf einer Webseite über HTTP können alle zwischengeschalteten Internetswitche und -betreibenden die gesamte Kommunikation mitlesen. Viele Webserver leiten heutzutage daher jede HTTP Anfrage sofort auf HTTPS weiter. Beispielsweise wird aus `http://tierbefreiung.de` dann `https://tierbefreiung.de` sodass der übertragene Inhalt verschlüsselt ist. Hierauf sollte sich jedoch nicht verlassen werden, zumal verschiedene Angriffe bedingt dafür sorgen können, dass eine solche Weiterleitung ausgesetzt wird (z.B. sog. „Downgrade-Angriffe“). Es empfiehlt sich daher unbedingt Firefox beizubringen HTTP-Verbindungen kategorisch abzulehnen und Ausnahmen ggf. nur manuell zu erlauben. Hierzu sollte das Add-On *HTTPS Everywhere* von der EFF (Electronic Frontier Foundation) installiert und so konfiguriert werden, alle unverschlüsselten Anfragen zu blockieren. Add-Ons können bei Firefox installiert werden, indem entweder auf Extras / Add-Ons geklickt wird oder indem in der Adressleiste `about:addons` eingegeben wird.

Um Aggregatoren und Tracker etwas einzuschränken gibt es mehrere Maßnahmen. Die erste ist es, einen Werbeblocker zu installieren. Nicht nur sind Werbebanner nervig, eklig und manipulativ, sie

dienen auch Aggregatoren / Trackern als Informationsquelle und können sogar Schadcode enthalten. Für Firefox kommen als effektive Werbeblocker „AdBlock Plus“ (ABP) und „AdBlocker Ultimate“ (ABU) in Betracht. Beide sind FOSS, haben jedoch verschiedene Vor-/Nachteile. ABU ist deutlich jünger, dafür aber kompromissloser und nicht wie ABP unternehmensfinanziert. Zusätzlich zu einem dieser beiden sollte auch Ghostery installiert und kurz konfiguriert werden. Das gleiche gilt für das Add-On namens Privacy Badger (ebenfalls von der EFF).

Weiter kann in den Firefox-Einstellungen unter „Datenschutz & Sicherheit“ eingestellt werden, wie „streng“ der integrierte Schutz sein soll. Die strikteste Einstellung führt in der Praxis kaum zu einer Einschränkung des gewünschten Funktionsumfangs und sollte daher standardmäßig gewählt werden. In den Einstellungen sollte auch unbedingt die Standardsuche geändert werden. Anstatt Google oder Bing sollte hier entweder StartPage oder DuckDuckGo eingestellt werden. StartPage anonymisiert jede Suchanfrage, verwendet jedoch unter der Haube die (technisch nun einmal exzellente) Google-Suchmaschine. Beide Suchmaschinen können als Add-On installiert werden oder manuell in der Such-Konfiguration von Firefox eingestellt werden.

All diese Maßnahmen sind jedoch nur eine Teillösung, wenn auch eine wichtige, insbesondere da sie so gut wie keine Einschränkung mit sich bringen. Ein noch wirksamerer Schutz, der jedoch einen kleinen Einrichtungsaufwand mit sich bringt ist das hauseigene Add-On „Firefox Multi-Account Containers“. Dieses erlaubt es verschiedene Webangebote hermetisch voneinander abzugrenzen, indem (beliebig viele) separate Container angelegt werden. Nach außen sieht jeder Container wie eine eigenständige Person aus. Es ließe sich beispielsweise ein Container für Onlineshops einrichten, ein Container für Social-Media, einer für Banking, usw. Der einzige Aufwand besteht darin, jede potenziell riskante Domain einem bestimmten Container zuzuweisen.

3.3 Das Zwiebelprinzip (Onion Routing) (T)

Eine zentrale Erkenntnis in Bezug auf Privatsphäre und Repressionsschutz ist, dass die wertvollsten Informationen für Angreifer*innen meist nicht der tatsächliche Inhalt übermittelter Nachrichten sind, sondern die Metadaten. Metadatum ist dabei ein Sammelbegriff. Er umfasst Daten darüber, wer mit wem spricht, wie oft, wann, wo, wie lange, über welche Kommunikationswege, welche Gruppen existieren und wer in diesen Mitglied ist, usw. Hieraus lassen sich nicht nur kommerzielle Interessen befriedigen („Wenn Dein Freund*innenkreis Produkt X mag, magst Du es vielleicht auch.“) sondern auch politische. Nicht umsonst versucht der deutsche Staat auf Drängen der EU seit über einem Jahrzehnt das Konstrukt der verpflichtenden massenhaften Vorratsdatenspeicherung durchzudrücken — bisher ohne großen Erfolg. Hier sollen nicht Telefonate aufgezeichnet werden, sondern „nur“ Informationen über die Gesprächsparteien.

Hiervor kann Kryptographie nicht schützen.

Kryptographie kann Inhalte verschlüsseln, was nur bedingt dabei hilft, Metadaten zu kaschieren (dazu mehr in den nächsten Abschnitten). Nichts hält den *Internet Service Provider* (ISP), davon ab, Protokoll darüber zu führen, mit welchen Servern ein Anschluss spricht, wie oft, welche Datenmengen fließen und so weiter. Und jeder Anschluss kann vom selben ISP über die intern vorliegenden Vertragsdaten mit den Realnamen der hinter dem Anschluss befindlichen Kund*innen in Verbindung gebracht werden. Das hängt mit dem grundlegenden Aufbau des Internets an sich zusammen, weniger mit dem Web. Eine oft vorgeschlagene Teillösung kann sein, im Internet nach anonymen Proxys zu suchen. Die Idee dahinter ist: Wir bauen eine verschlüsselte Verbindung zu einem bestimmten Server (Proxy) im Internet auf und schicken *sämtliche* Daten über diesen. Da dies nicht eine Person tut, sondern sehr viele,

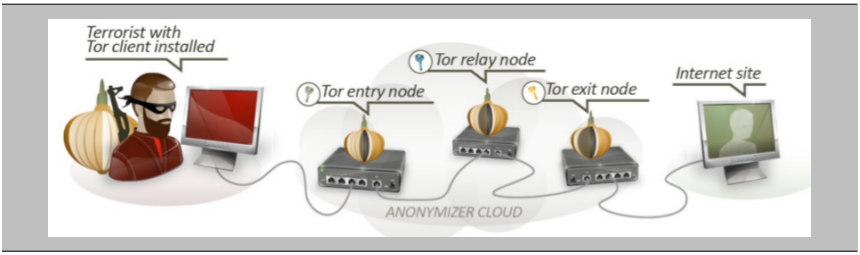


Abbildung 5: „Tor Stinks“ (NSA). Quelle: Edward Snowden [8]

können Außenstehende nicht mehr so leicht Metadaten extrahieren. Durch ordentliche Verschlüsselung per HTTPS kann der Proxy nicht unsere Daten mitlesen. Das Problem: Proxys zu finden, die sowohl vertrauenswürdig sind wie dauerhaft stabil laufen, ist so gut wie unmöglich.

Die Lösung: *Das Tor-Netzwerk*. Bei Tor (ursprünglich von: The Onion Router) ist eine Verbindung wie die Schalen einer Zwiebel aufgebaut und dient der Anonymisierung. Anstatt dass Alice direkt mit Bob spricht, spricht Alice mit Anna, Anna mit Arthur, Arthur mit Ali und Ali dann mit Bob. Nur, dass in dieser Kette Anna, Arthur und Ali bei jeder neuen Verbindung zufällig aus allen Benutzer*innen des Tor-Netzwerks gewählt werden. Und jede Verbindung ist separat von Alice vorverschlüsselt: Jede Station auf dem Weg nimmt je eine „Verschlüsselungsschale“ von der Nachricht weg, so dass die Daten die von jedem Stopp weitergeleitet werden, anders aussehen als die, die sie empfangen haben. Hierdurch und durch ein paar weitere Tricks wird verhindert, dass eine Nachricht durch das Netzwerk verfolgt werden kann, ob für Außenstehende oder für die Stationen auf dem Weg. Und natürlich, dass irgendwer außer Bob den Nachrichteninhalt sehen kann.

Der Clou: Aus Sicht von Bob (zum Beispiel ein Webserver) sieht das Ganze so aus, als wäre die Anfrage von Ali gekommen (in diesem Szenario ein sogenannter „Tor-Exit Knoten“). Bei der nächsten Ver-

bindung wird jedoch ein anderer Exit-Knoten gewählt. Hierdurch werden so viele Spuren erzeugt, dass nicht nachvollziehbar ist, was die echte Spur ist. In Kombination mit den oben genannten Maßnahmen kann hierdurch ein extrem hoher und verlässlicher Schutz entstehen, der (mit etwas Selbstdisziplin) Repression so gut wie unmöglich macht. Vorausgesetzt immer, das System, von dem aus gearbeitet wird, ist nicht kompromittiert.

Obwohl wir für diese Broschüre eine korrektere/informativere Grafik hätten erstellen können, ist die Darstellung der NSA aus einer von Snowden veröffentlichen Präsentation namens „Tor stinks“ (Tor stinkt) [8] zu unterhaltsam, um sie nicht zu verwenden (Abbildung 5).

3.4 Eine ziemlich sichere Verwendung des Webs (P)

Das Tor-Projekt stellt einen fertig für die Benutzung mit dem Tor-Netzwerk konfigurierten Firefox zur Verfügung. Dieser ist leider nicht direkt aus den Debian (geschweige denn Ubuntu) Paketquellen erhältlich. Bei Debian gibt es jedoch das Meta-Paket namens `torbrowser-launcher`, mit dem sich der Tor-Browser semi-automatisch herunterladen und installieren lässt. Alternativ kann er auch von `https://torproject.org/download` manuell heruntergeladen werden. So oder so muss der Tor-Browser unabhängig von der Systemaktualisierung auf dem aktuellsten Stand gehalten werden. Alternativ kann der Tor-Browser auch von den dafür bereitgestellten Paketquellen des Tor-Projekts installiert werden. [9]

In der Praxis ist der Gebrauch des Tor-Browsers etwas langsamer als der unmodifizierte Firefox. Wenn möglich sollte auch vermieden werden, das Tor-Netzwerk mit zu viel unnötigem Multimedia-Inhalt zu überlasten. Wer also nicht ausgerechnet HD-Filme streamen möchte, kann mit dem Tor-Browser anonym und sicher im Web surfen. Allerdings kann auch Tor natürlich nicht verhindern, dass sich ein Mensch durch freiwillige Angabe von persönlichen Daten selbst identifiziert.

4 Internetkommunikation

4.1 Grundlagen Asymmetrischer Kryptographie (T)

Im Gegensatz zu symmetrischen Verfahren setzen *asymmetrische* Verfahren auf Eigenschaften Diskreter Mathematik (siehe Kasten *Mathematische Grundlagen asymmetrischer Kryptographie* und [10]). Von zentraler Bedeutung ist, dass in asymmetrischen Verfahren initial zwei separate Schlüssel generiert werden, wie Abbildung 6 zeigt. Diese beiden Schlüssel werden zwar zufällig erzeugt, hängen jedoch mathematisch zusammen: Was mit einem der beiden verschlüsselt wird, kann nur mit dem anderen Schlüssel wieder entschlüsselt werden. Ein solches Schlüsselpaar wird üblicherweise so verwendet, dass eines der beiden Schlüssel, der sogenannte *private Schlüssel*, auf alle Fälle geheim bleiben muss, während der andere, *öffentliche Schlüssel* wild in der Welt verteilt werden darf. Alle dürfen diesen öffentlichen Schlüssel bekommen, auch Personen, denen null vertraut wird.

Alice möchte nun wie zuvor einen Text an Bob schicken. Dann kann Alice diesen mit Bobs öffentlichem Schlüssel verschlüsseln und den daraus entstandenen Krypttext über eine beliebige ungesicher-

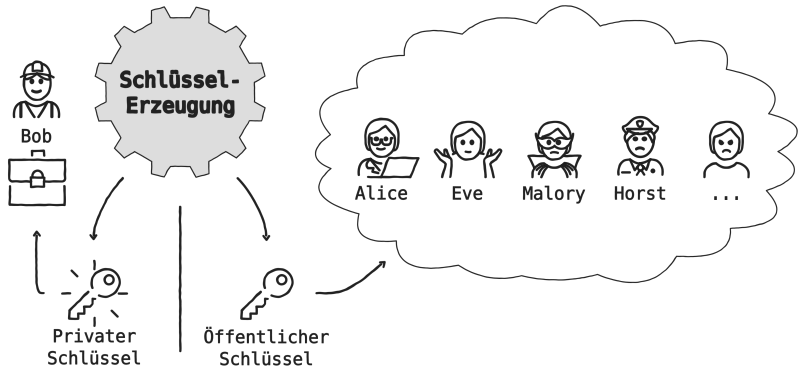


Abbildung 6: Erzeugung asymmetrischer Schlüssel

Mathematische Grundlagen asymmetrischer Kryptographie

Für interessierte Lesende soll hier eine (sehr) kurze und möglichst barrierearme Skizze der Mathematik gegeben werden, welche modernen asymmetrischen Verfahren zugrundeliegt. Es gibt eine Reihe wichtiger mathematischer Funktionen, auf die asymmetrische Kryptoverfahren aufsetzen. Ihnen ist gemeinsam, dass die Funktion in eine Richtung auszurechnen einfach ist, während diese umzukehren so schwer ist, dass es in der Praxis als unmöglich gilt.

RSA beispielsweise beruht auf der Härte von Primfaktorzerlegung: Multiplikation von Primzahlen ist einfach, Zerlegung in diese jedoch nicht. Primzahlen sind dabei Zahlen welche durch genau 1 und sich selbst teilbar sind. Zusätzlich hat jede positive ganze Zahl eine einzige eindeutige Zerlegung in Primfaktoren. Etwa 12 besitzt die Zerlegung $2 \cdot 2 \cdot 3$. Die Zahl 145426 zerfällt in die Primzahlen $2 \cdot 19 \cdot 43 \cdot 89$.

Für zwei große, zufällige Primzahlen p, q können Zahlen e und d gefunden werden, sodass für jeden Klartext m, der Krypttext $c = m^e$ berechnet werden kann. Dieser kann dann mit c^d wieder zu m entschlüsselt werden. Sind solche Zahlen gefunden, wird e als öffentlicher Schlüssel verwendet und d als privater Schlüssel. Auf den verwendeten Modulus $n = p \cdot q$ kann hier aus Platzgründen nicht eingegangen werden. Für eine detaillierte Einführung in die Funktionsweise sei hier auf einschlägige Quellen verwiesen (etwa [10]).

te Leitung an Bob schicken, zum Beispiel über das Internet (Abbildung 7). Und das ohne dass Eve etwas damit anfangen kann, obwohl Eve ebenfalls im Besitz von Bobs öffentlichem Schlüssel ist. Nur Bob mit dem dazu passenden privaten Schlüssel kann den Krypttext wieder entschlüsseln. Allerdings kann sich Bob zunächst nicht sicher sein, dass die Nachricht tatsächlich von Alice stammt, denn die ganze Welt kennt den öffentlichen Schlüssel und hätte die Nachricht fälschen können. Um dieses Problem zu lösen, wird sich zunutze gemacht, dass die Rolle eines privaten und öffentlichen Schlüssels umgekehrt werden kann: Was mit dem öffentlichen verschlüsselt wurde, kann mit dem privaten wieder entschlüsselt werden, aber auch was mit dem privaten „verschlüsselt“ wurde, kann später mit dem öffentlichen „entschlüsselt“ werden. Wenn Alice

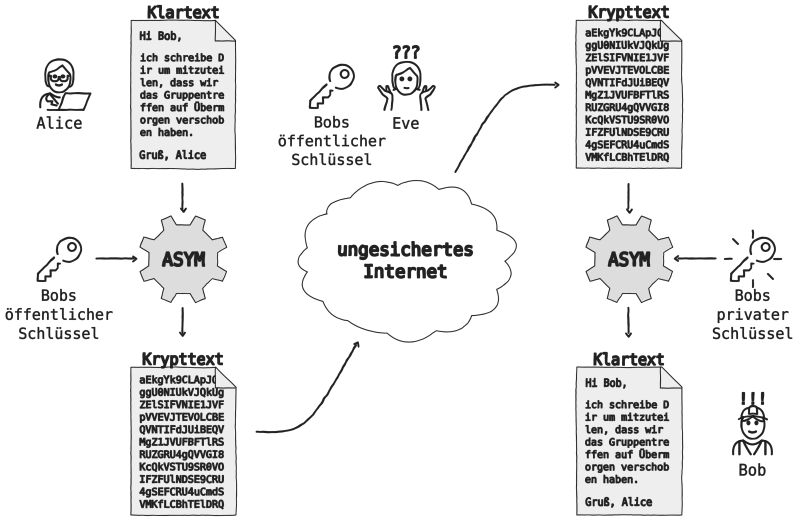


Abbildung 7: Aufbau asymmetrischer Kryptographie

also auch ein eigenes Schlüsselpaar erzeugt und ebenfalls den privaten Schlüssel niemandem verrät, aber den öffentlichen an u. a. Bob gibt, kann Alice die Nachricht *signieren*. Nur Alice kann eine Signatur erzeugen, die mit Alices öffentlichen Schlüssel entschlüsselt einen bestimmten Wert ergibt.

Doch der Klartext sollte besser nicht direkt für die Signatur verwendet werden, sonst könnten alle mit Alices öffentlichen Schlüssel – also alle – diesen lesen. Mit einer sogenannten *Hashfunktion* kann aus einem beliebigen Text eine Zahl gemacht werden, die auf keine Weise wieder in den ursprünglichen Text überführt werden kann – eine Art Einbahnverschlüsselung. Diese Zahl wird als *Hash* bezeichnet. Verschlüsselt Alice also den Hash des zu übertragenden Klartextes mit dem eigenen privaten Schlüssel, kann Bob, nach Entschlüsselung des Krypttextes, überprüfen dass die Nachricht tatsächlich von Alice kam, indem Bob die Signatur mit Alices öffentlichem Schlüssel „entschlüsselt“ und mit dem Hash abgleicht.

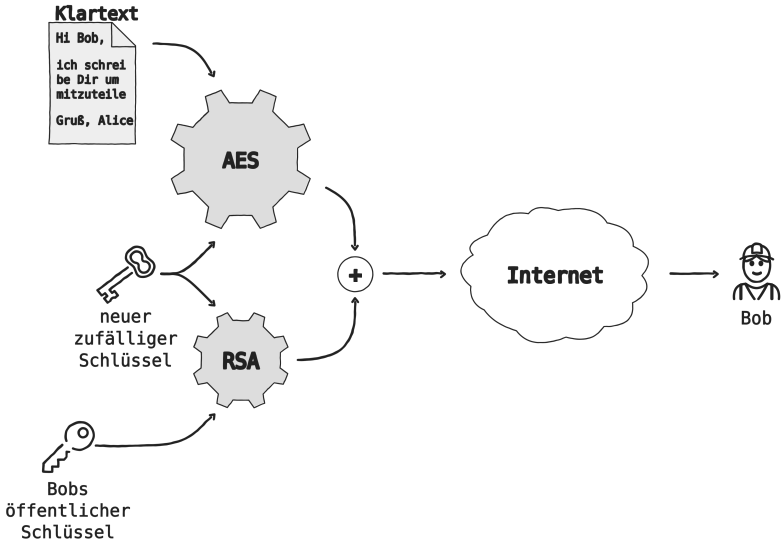


Abbildung 8: Temporäre Schlüsselerstellung in OpenPGP

Wichtige asymmetrische Verfahren sind RSA, ECC und ElGamal. Sie haben gemeinsam, dass ihre Verwendung deutlich aufwändiger (heißt: langsamer) ist als symmetrische Verfahren, sodass sie in der Praxis zusammen mit symmetrischen Verfahren verwendet werden, meist mit dem rasend schnellen AES oder IDEA. Beispielsweise kann für jede Nachricht (sehr schnell) ein temporärer AES-Schlüssel erzeugt werden, mit dem dann der Klartext verschlüsselt wird. Dann übernehmen RSA und Co nur die Aufgabe, diesen temporären AES-Schlüssel mit dem öffentlichen Schlüssel Bobs (also dem der empfangenden Person) zu verschlüsseln — siehe Abbildung 8.

Während bei AES Schlüssellängen von 128 bis 256 bit (Dezimalzahl mit 39 bzw. 78 Stellen) verwendet werden, werden grob alle Jahrzehnte die empfohlenen Schlüssellängen für asymmetrische Verfahren verdoppelt, um mit Angriffen mit immer besser werdender

Hardware mithalten zu können. Derzeit sind für RSA Schlüssellängen von 3.072 bis 4.096 Bits empfehlenswert (rund 1.000 Dezimalstellen).

4.2 Email (T)

Einige Dekaden älter als das Web, ist Email bis heute integraler Bestandteil digitaler Kommunikation. Das System stammt dabei erst recht aus einer Ära in der Sicherheit kein relevantes Designkriterium darstellte. Um aus diesem prinzipiell absolut unsicheren System ein sicheres zu machen, müssen wir zunächst zwei Dinge verstehen: Wie funktioniert das Emailsystem und gegen welches Angriffsmodell wollen wir uns verteidigen? Um eine Email von Alice zu Bob zu befördern sind eine Menge Computer beteiligt, wie die vereinfachte Abbildung 9 zeigt.

Auf dem Wege zwischen Alices Laptop zu Bob muss die Nachricht mehrfach übertragen und gespeichert werden, unter anderem weil Email asynchron ist: Weder Bob noch Bobs Computer müssen online sein, damit Alice eine Email verschicken kann und gleichzeitig kann Alice den Computer ausschalten, bevor die Nachricht bei Bob

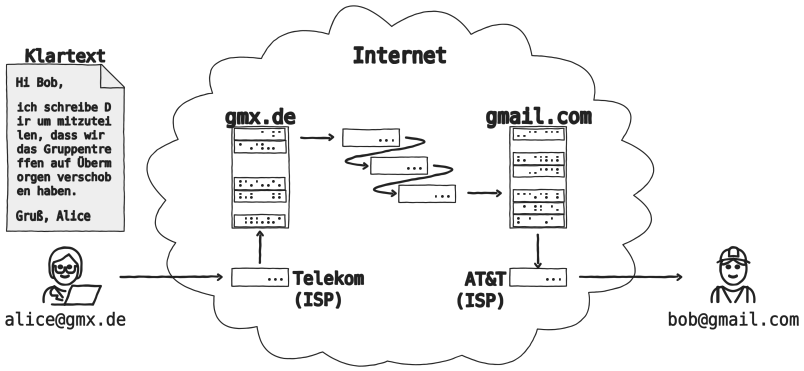


Abbildung 9: Übertragungspfad einer Email

eingetroffen ist. Sämtliche Pfeile dort stellen eine unverschlüsselte Übertragung dar und jeder Computer auf diesem Weg sowie Angreifer*innen dazwischen können Inhalt und Metadaten mitlesen.

Darum setzt sich, analog zum Web, bei der Übertragung von Emails immer mehr durch, dass Betreiber*innen Transportverschlüsselung über TLS anbieten. Das bedeutet, dass die Anmeldung bei und Übertragung an Alices Emailhoster (im Beispiel: gmx.de) verschlüsselt wird – hierzu muss das Emailprogramm so eingerichtet werden, dass es die Kommunikation zum Server verschlüsselt. Dadurch kann auf der Verbindung (im Beispiel: Telekom) keine nennenswerte Datensammlung durchgeführt werden, außer dass Alice anscheinend die Emaildienste von gmx.de in Anspruch nimmt. Ganz genauso kann auf der Empfangsseite Transportverschlüsselung aktiviert werden. Manche Anbieter*innen werben sogar damit, dass sie die Verbindung zu Servern anderer Email-Anbieter*innen verschlüsseln.

Dieses Modell würde ausreichen, wenn wir volles Vertrauen hätten, dass sowohl beide Email-Anbieter*innen als auch die Empfänger*innen bei der Einrichtung keine Fehler gemacht haben sowie, dass sie nicht absichtlich Daten weitergeben. Zumindest für beide oben genannten Anbieter*innen wissen wir jedoch, dass sie in der Vergangenheit mit Geheimdiensten/Regierungen kollaboriert haben und es vermutlich immer noch tun (Siehe Abbildung 1 und unbedingt [11]). Wer also GMX oder Gmail für sensitive Projekte nutzt, sollte sich das vielleicht nochmal genauer überlegen.

4.3 OpenPGP (T)

Auf das bestehende Emailsystem lässt sich das Ende-zu-Ende Verschlüsselungsverfahren namens *PGP* aufsetzen (genauer: OpenPGP). Dieses verwendet eine Kombination aus symmetrischer und asymmetrischer Kryptographie, um zu gewährleisten, dass niemand außer Alice und Bob den Inhalt und (inzwischen auch) die Betreffzeile der Email lesen können. Die Metadaten vermag allerdings

auch PGP nicht zu schützen. Um auch diese zu schützen müssen PGP, Transportverschlüsselung sowie vertrauenswürdige Serverbetreibende vorhanden sein.

Für die Verwendung von PGP-verschlüsselten Emails müssen Alice und Bob jeweils RSA oder ECC Schlüsselpaare anlegen und jeweils die öffentlichen Schlüssel sicher austauschen. Der Austausch der öffentlichen Schlüssel entpuppt sich dabei häufig als Henne-Ei-Problem, insbesondere wenn keine persönliche Leitung besteht, über die die Authentizität der anderen Person verifiziert werden kann: Um eine sichere Leitung aufzubauen brauchen wir initial eine sichere Leitung.

Ein öffentlicher Schlüssel kann zum Beispiel auf frei verfügbare, sogenannte Schlüsselserver hochgeladen werden, etwa auf [12]. Eine Person, die mit einer unbekannten Person verschlüsselt kommunizieren möchte, kann dann diesen Schlüssel von dem Server herunterladen und installieren. Da dies aber völlig mühelos beliebig gefälscht werden kann – jede*r kann für jede beliebige Emailadresse einen Schlüssel erstellen und diesen hochladen – muss stets die Authentizität sichergestellt werden.

Zur Überprüfung verfügen alle öffentlichen Schlüssel über einen sogenannten Fingerabdruck/Fingerprint: Eine kurze Zeichenfolge, die über ein Medium ausgetauscht werden kann, welches nicht zwingend abhörsicher aber verfälschungssicher sein muss. Beispielsweise eine Telefonleitung, wenn die Stimme des Gegenübers bekannt ist. Oder ein Printmedium; mein Fingerabdruck ist etwa 5372 25DB 4CF0 9C5B 5FB2 22B4 667D EF4A 0888 50E4.

4.4 Email mit Ende-zu-Ende Verschlüsselung (P)

Zur Einrichtung Ende-zu-Ende-verschlüsselter Email muss zuerst ein verlässliches Emailprogramm installiert werden. Hier wird das FOSS-Programm Thunderbird empfohlen, welches über die Debian Paketquellen installierbar ist. Abhängig von der installierten

Version muss zusätzlich noch ein Add-On namens Enigmail installiert werden, wenn Thunderbird 68 oder älter installiert ist. Ab Thunderbird 78 ist PGP nativ implementiert und das Add-On wird nicht mehr benötigt. Zusätzlich zu Thunderbird kann noch das deutsche Sprachpaket über die Paketverwaltung oder den Add-On-Manager installiert werden. Da Thunderbird ein Schwesterprojekt des Web-Browsers Firefox ist, läuft die Installation von Add-Ons hier analog zu Abschnitt 3.2.

Über die Schlüsselverwaltung lassen sich neue Schlüsselpaare erzeugen, erhaltene Schlüssel importieren und Fingerabdrücke überprüfen. Bei der Erstellung eines Schlüsselpaares kann eine sogenannte *Passphrase* verwendet werden, mit der der private Schlüssel selbst symmetrisch verschlüsselt wird, bevor er auf der Festplatte gespeichert wird. Das führt dazu, dass vor der Benutzung des privaten Schlüssels (etwa zur Entschlüsselung einer Email) die Passphrase eingetippt werden muss — manuell oder durch einen Passwordmanager. Dieser hohe Aufwand ist nicht notwendig, wenn das System auf dem der Schlüssel gespeichert ist, ordentlich abgesichert ist, wenn also wie oben ausgeführt mindestens das home-Verzeichnis verschlüsselt ist und auch sonst keine Löcher ins System geschlagen wurden.

4.5 Off-The-Record (OTR) (T)

OTR [13] ist ein Protokoll, das noch höhere Garantien bietet als der von beispielsweise PGP angebotene Sicherheitsgrad. Ursprünglich als synchrones Verfahren entworfen, bei dem also alle Parteien, die sicher miteinander kommunizieren wollen, zeitgleich online sein müssen, gibt es inzwischen auch Versionen, die asynchrone Kommunikation ermöglichen. Die Idee hinter OTR ist es, beliebig auf bestehende – unsichere – Chatsysteme aufgesetzt werden zu können.

Es garantiert, neben der auch von PGP angebotenen Verschlüsselung und Authentifizierung, die *Plausible Abstreitbarkeit* im Nach-

hinein und sogenannte *Perfect Forward Secrecy* (PFS). Erstere ermöglicht es stets abstreiten zu können, eine Nachricht verfasst zu haben, da das Protokoll so aufgebaut ist, dass *im Nachhinein* jede*r die Möglichkeit hat, beliebig Nachrichten zu fälschen. PFS hingegen ermöglicht, dass, auch wenn in der Zukunft ein Angriff auf die privaten Schlüssel gelingt, niemals nachträglich eine möglicherweise aufgezeichnete Unterhaltung entschlüsselt werden kann. Dies gelingt, indem für jede Unterhaltung ein neuer Sitzungsschlüssel erzeugt wird, ohne dass der permanente private Schlüssel direkt für die Übertragung dieses Sitzungsschlüssels verwendet wird[6].

4.6 Messenger (P)

Als schnellere Alternative zu Email, insbesondere in Zeit ubiquitärer „Smart"phones, sind Messengerdienste immer beliebter. Es liegt jedoch in der Natur eben dieser Geräte, dass sie immer wieder Sicherheitslücken aufweisen und außerdem, dass sie viel leichter in die Hände eventueller Angreifer*innen fallen können. Selbst ein ideales, perfektes Kommunikationssystem, welches auf einem kompromittierten System läuft, kann niemanden vor Angriff schützen. Es gibt aber einige wichtige Sicherheitsoptionen, welche die Kapazität haben, die Risiken eines Angriffs zu minimieren.

Die Klasse der Messenger zeichnet sich dadurch aus, dass sie, so wie auch Email, asynchrone Kommunikation ermöglichen; dass also nicht alle kommunizierenden Parteien gleichzeitig online sein müssen. Diese Architektur erfordert *notwendigerweise* das Hinzunehmen einer Serverinfrastruktur, welche die Nachricht speichert, bis sie von der empfangenden Person abgeholt werden kann („Store and Forward"). Die Vertraulichkeit und Integrität von verschiedenen Messengern ist daher grundsätzlich von der Vertrauenswürdigkeit der Softwareimplementation sowie der Serverbetreibenden abhängig: Bietet der Messenger Ende-zu-Ende-Verschlüsselung?

[6] Genauer: Der private Schlüssel wird nur zur Signatur des Diffie-Hellman-Austauschs verwendet.

Ist sowohl Messengersoftware wie auch Serversoftware FOSS? Kann sie also auf Fehler und Hintertüren untersucht werden? Welche Informationen sind für die Betreibenden sichtbar (Stichpunkt Metadaten)? Unter diesen Gesichtspunkten können wir uns einige populäre Messenger anschauen.

4.6.1 WhatsApp

Der derzeit noch populärste, WhatsApp, scheitert an den meisten dieser Hürden katastrophal. Vor einigen Jahren stellte Facebook zwar ihren WhatsApp-Dienst auf ein unabhängiges Protokoll um (dazu unten mehr), welches Ende-zu-Ende-Verschlüsselung à la OTR erlaubt, jedoch ist WhatsApp komplett geschlossen: Sowohl die App wie auch die Serversoftware sind nicht quelloffen. Wenig überraschend ist auch, dass WhatsApp mehrfach dabei erwischt wurde, massiv Daten von den Benutzer*innen abzugreifen, darunter das gesamte Adressbuch, und für Facebooks kommerzielle und politische Zwecke zu verwenden. [14, 15] Die Tatsache, dass WhatsApp zusätzlich mit dem Tierversuchsgehilfen AirFrance/KLM zusammenarbeitet [16], macht es erst recht einfach, diesen unzuverlässigen Dienst einfach nicht mehr zu benutzen.

4.6.2 Telegram & Threema

Wer sich nicht so gerne von der NSA abhören lässt, kann sich stattdessen auch den russischen FSB, die KGB-Nachfolgeorganisation, aussuchen. [17, 18] Der russische Messenger Telegram verwendet ein semi-kaputtes Ende-zu-Ende-Protokoll, welches manche Nachrichten verschlüsselt, andere nicht, und läuft überdies über unfreie Serversoftware. Mit Telegrams „„„Bereitschaft““" die russischen Strafverfolgungsbehörden beim „Kampf gegen Extremismus“ zu unterstützen dürfte klar sein, dass auch hier zumindest Metadaten ausgewertet werden.

Etwas solider sieht der kostenpflichtige Messenger Threema aus der Schweiz aus, der ebenfalls Ende-zu-Ende-Verschlüsselung unterstützt. Seit Ende 2020 ist der Threema Client unter einer FOSS

Lizenz veröffentlicht worden, die Serverseite jedoch nicht. In Bezug zu letzterer ist keine Bestrebung zur Öffnung zu erkennen. Wir müssen uns in Bezug auf die Zuverlässigkeit also auf einen einzigen Audit verlassen, der die Software untersuchen durfte. [19] In welchem Umfang Threema Metadaten abgreift ist wenig untersucht, insbesondere weil der Messenger wenig Verbreitung findet.

4.6.3 Signal

Vieles richtig macht hingegen der offene Messenger Signal, der im Gegensatz zu oben genannten von einer non-profit-Initiative (Signal Foundation, vormals Open Whisper Systems) entwickelt wird. Signal hat die auf OTR basierende freie Ende-zu-Ende-Verschlüsselung entwickelt und verschlüsselt konsequent alle Daten. Telefonnummern werden nur gehasht zum Kontaktaufbau auf dem Server gespeichert, der ebenfalls quelloffen ist. Im Jahr 2016 erhielt Open Whisper Systems eine Zwangsvorladung vom FBI, in der sie gezwungen wurden, sämtliche Daten über zwei Personen einem Gericht offenzulegen [20]: Aufgrund der durchgängigen Verschlüsselung waren die einzigen Daten, auf die die Betreibenden Zugang hatten, das Registrierungsdatum und das Datum des letzten Verbindungsaufbaus. Weder Kontaktinformationen, noch Regelmäßigkeit der Benutzung etc. konnten sie preisgeben, selbst wenn sie gewollt hätten.

Der Messenger wartet mit einigen interessanten Funktionen auf; sämtliche Daten auf dem Smartphone werden nur gespeichert abgelegt und sind nur durch Eingabe eines Passworts verfügbar (dies muss in einigen Versionen explizit aktiviert werden). Benutzende können durch nebeneinanderhalten zweier Telefone auf einfache Art und Weise die Fingerabdrücke ihrer öffentlichen Schlüssel vergleichen. Zusätzlich können Unterhaltungen so eingestellt werden, dass sie nach einer vorgegebenen Zeit verschwinden. D.h., alle Personen können sich sicher sein, dass nach der eingestellten Zeit die übermittelten Daten auf keinem Gerät mehr gespeichert sind. Es verfügt über einen Mechanismus der erlaubt Zensurbemühungen

zu umgehen, wie etwa in Ägypten und den Emiraten, wo Signal seit 2016 (erfolglos) blockiert wird. [21] Im Rahmen der massiven Repression der US-Polizei gegen die BLM-Bewegung (Black Lives Matter) implementierte Signal Mitte 2020 zum Support von BLM einen automatischen Gesichtsfilter. [22, 23]

Ungeachtet dieser äußerst hochwertigen Sicherheitsvorkehrungen sollte nach der Installation der Signal-App einmal durch die Privatsphäre Einstellungen gegangen werden und sichergestellt werden, dass die relevanten Einstellungen aktiviert sind.

Es gibt keinen guten Grund, weiterhin kaputte oder unvollständige Messenger wie WhatsApp zu benutzen, insbesondere da Signal immer wieder beweist, die Privatsphäre der Benutzer*innen ernst zu nehmen. Ein paar Nachteile hat allerdings auch Signal; es erfordert derzeit mit einer echten Mobiltelefonnummer verknüpft zu sein und daher keine vollständige Anonymität – die Signal Foundation hat jedoch mehrfach bekundet daran zu arbeiten, Signal zukünftig auch ohne Bindung an eine Telefonnummer anzubieten. Zusätzlich läuft das Protokoll über einen einzigen Serveranbieter (nämlich die Signal Foundation), wodurch diese gezielt angegriffen werden kann.

4.6.4 Matrix & Jabber

Alternativen zu Signal mit vergleichbaren Sicherheitsleveln sind etwa der auf dem Matrix-Protokoll basierende Element Messenger (früher Riot) und das (technologisch noch bessere) synchrone Chatprotokoll Jabber/XMPP+OTR. Beide sind dezentralisierte/föderierte Protokolle, bei denen sich Nutzer*innen einen Server aussuchen oder sogar einen eigenen aufsetzen, trotzdem aber miteinander kommunizieren können. Leider sind diese aber nicht sehr verbreitet.

5 Zusammenfassung & Ausblick

5.1 Tails (P)

Tails ist das zweite Betriebssystem, dessen Einführung am Anfang dieses Textes versprochen wurde. Es ist eine auf Debian basierende GNU/Linux-Distribution, deren Fokus eine kompromisslose Privatsphäre und Sicherheit ist, und wird – zurecht – empfohlen von der EFF, dem Tor-Projekt (welches Tails auch finanziell unterstützt), Edward Snowden und dem Riseup-Kollektiv. Die Tails-Distribution ist dafür konzipiert als live-System von z.B. einem USB-Stick gestartet zu werden, insbesondere vergisst es sämtliche Einstellungen und Daten nach dem Herunterfahren. Sämtlicher TCP-Internetverkehr wird standardmäßig durch Tor getunnelt und es kommt vorinstalliert mit Thunderbird mit PGP, dem für OTR-Chat konzipierten Pidgin-Chatprogramm und einigen weiteren Hilfsprogrammen. Tails kann von der offiziellen Webseite [24] heruntergeladen werden und ist selbstverständlich Free and Open Source Software.

Obwohl es für den alltäglichen Einsatz nicht unbedingt den Komfort von etwa Debian bietet, ist es ein unermesslich wichtiges Werkzeug zur sicheren und anonymen Arbeit von Aktivist*innen und Journalist*innen weltweit.

5.2 Fazit

Computersicherheit geht uns alle an: es ist nicht nur Selbstschutz, sondern auch Schutz unserer Freund*innen, Genoss*innen, Mitstreiter*innen und Kompliz*innen. Wie dieser Text hoffentlich zeigen konnte, gibt es vielfältige Möglichkeiten unsere Privatsphäre und unsere Sicherheit zu schützen. Sei es vor der kapitalistischen Verwertung durch Internetgiganten oder vor dem repressiven Zugriff des deutschen Mörderstaates. Privatsphäre steht häufig der eigenen Bequemlichkeit entgegen, da es Arbeit kostet, sich hiermit auseinanderzusetzen und sichere Systeme zu warten. Doch es ist von zentraler Bedeutung, dass wir einen verantwortungsvollen

Umgang mit digitaler Sicherheit pflegen. Das ist wie Maske-tragen, es kann nerven, aber es kann uns und andere vor viel Schlimmerem schützen. Und kein vernünftiger Mensch würde ja eine kleine Unannehmlichkeit verweigern, die anderen und sich selbst das Leben retten kann, richtig?

Da sich aufgrund des Umfangs und Themas dieses Textes möglicherweise die Notwendigkeit für Korrekturen ergeben kann, findet sich neben den Referenzen in anklickbarer Form, auch eine Auflistung mit Ergänzungen und Errata unter [25]. Feedback und Korrekturen werden sehr gerne empfangen unter `alan_tierbefreiung@riseup.net`, natürlich am liebsten PGP-verschlüsselt.

Quellen

[1] www.washingtonpost.com/wp-srv/special/politics/prism-collection-documents/

[2] search.edwardsnowden.com/docs/MakingThingsMeasurable TechnologyTrendingChallengesandApproaches2014-12-28_nsadocs_snowden_doc

[3] www.redhat.com/de/topics/open-source/what-is-open-source

[4] www.gnu.org/philosophy/free-sw.de.html

[5] www.debian.org/index.de.html

[6] wiki.debianforum.de/Installation_für_Einsteiger

[7] wiki.ubuntuusers.de/Partitionierung/Grundlagen/

[8] edwardsnowden.com/wp-content/uploads/2013/10/tor-stinks-presentation.pdf

[9] support.torproject.org/de/apt/tor-deb-repo/

[10] www.inf.hs-flensburg.de/lang/krypto/protokolle/rsa.htm

[11] Linus Neumann. 30C3 - Bullshit made in Germany. media.ccc.de/c/30c3 oder youtu.be/p56aVppK2W4

[12] keys.openpgp.org

[13] otr.cypherpunks.ca

[14] netzpolitik.org/2016/abschied-von-whatsapp-fuenf-gute-gruende-fuer-den-messenger-wechsel/ (von 2016 aber großteils noch aktuell)

[15] www.heise.de/news/WhatsApp-Co-Forscher-warnen-vor-massenhaftem-Auslesen-von-Kontakten-4904618.html

[16] news.klm.com/klm-first-airline-with-verified-whatsapp-business-account/

[17] www.theverge.com/2018/3/20/17142482/russia-orders-telegram-hand-over-user-encryption-keys

[18] www.independent.co.uk/news/world/europe/telegram-russia-ban-lift-messaging-app-encryption-download-a9573181.html

[19] threema.ch/press-files/2_documentation/security_audit_report_threema_2019.pdf

[20] signal.org/bigbrother/eastern-virginia-grand-jury/

[21] netzpolitik.org/2016/messenger-app-signal-umgeht-sperre-in-aegypten/

[22] www.nytimes.com/2020/06/11/style/signal-messaging-app-encryption-protests.html

[23] www.cyberscoop.com/george-floyd-protest-phone-security/

[24] tails.boum.org/index.de.html

[25] bit.ly/3kcB6Lv (schreibgeschütztes riseup-Pad, mindestens verfügbar bis Mai 2022)

Nicht so empowernd wie Berichte von befreiten Tieren, blockierten Mordfabriken oder aus der Nase blutenden Nazis. Aber ebenso wichtig wie zuvor Genanntes: Digitale Sicherheit und der damit verbundene Schutz vor Repression. Und wer jetzt sagt, „ich habe ja nichts zu verbergen" muss sich mindestens zwei Gegenfragen gefallen lassen: 1. „Wieso nicht? Es wird mal Zeit!" und 2. „Sicher?"

Die Kontrolle von Daten ist das primäre Werkzeug repressiver Organe und der treibende Motor kapitalistisch motivierter Verwertungslogik. Daher soll hier eine Grundlage geschaffen werden, die es ermöglicht, die Funktionsweise von sowohl Überwachungsmechanismen (im weitesten Sinne) wie auch unserer Gegenmaßnahmen zu verstehen. Was bedeutet es eigentlich, wenn wir heutzutage von Abhörung und Überwachung sprechen? Was ist Kryptographie und welche Bedeutung kommt ihr zu?

Dieser Text richtet sich vornehmlich mit Theorie und Praxis an Laien, doch meine Hoffnung ist es, auch für fortgeschrittene Lesende neue Perspektiven zu schaffen.

2021