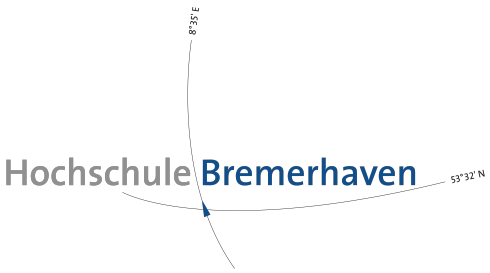


Lecture Notes

Einführung in die IT-Sicherheit

Wintersemester 2020/21

27. Januar 2022

Prof. Dr. Lars Fischer
lars.fischer@hs-bremerhaven.de

License

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

In informal terms that means that you are free to reuse and adapt this work in part or in whole as long as you

a) give appropriate credit to the author (me),

b) distribute your deriviate work under the same license.

You are free to contact me for a different license if you have good reasons why this license is too restrictive for your purpose.

Contact me via email at lars.fischer@hs-bremerhaven.de.

Inhaltsverzeichnis

seit 2020

- Hochschule Bremerhaven, IT-Sicherheit

2015 bis 2020

- GL/Wiss. MA: OFFIS
- Wiss. MA: Carl-v.-Ossietzky Universität
  - Cyber-Resilient Architectures and Security
  - Adversarial Resilience Learning

vor 2015 Siegen: Wiss. MA und Prof. IT-Sicherheit

- Location Privacy, Soziale Netzwerke

vor 2011

- Berater IT-Sicherheit
- Promotion: TU-Darmstadt
  - Privatheitsmetriken, Fahrzeugnetze

1

Syllabus

Folgend werden die Rahmenbedingungen des Kurses IT-Sicherheit an der Hochschule Bremerhaven im Wintersemester 2021/22; Organisation, Lernform und Prüfungsform beschrieben.

> Zur Motivation:
>
> "Den besten Teil unserer Lebenszeit verbringen wir auf der Arbeitsstelle. Man muss deshalb lernen, so zu arbeiten, dass die Arbeit leicht und zu einer ständigen Lebensschule wird."

Der Kurs *IT-Sicherheit* ist auf eine konstante Beteiligung an allen Teilen des Kurses ausgelegt. Wesentliche Prüfungsleistungen werden im laufenden Semester erbracht. Es ist deshalb weder möglich noch sinnvoll den Kurs als Blockkurs durchzuführen. *Bestehen ernsthafte Hindernisse die einer konstanten Beteiligung entgegenstehen müssen diese frühzeitig angemeldet werden!* Eine nachträgliche Anpassung der Lern- und Prüfungsbedingungen ist nicht vorgesehen.

- Syllabus / Labor
- IT-Sicherheit Grundbegriffe
- Sicherheitsarchitektur
- Access Control
- Kryptographie
- Distributed Authentication
- Kommunikationssicherheit
- Trust
- Digitales Geld

- Sichere Software
- Anonymity/Unlinkability
- Security Management

Voraussetzungen

Die Veranstaltung IT-Sicherheit baut auf Wissen aus vorherigen Veranstaltungen im Modulhandbuch auf. Damit Sie sich besser auf die Teilnahme vorbereiten können, finden Sie hier eine Auflistung von wichtigen Themen aus früheren Modulen.

- TCP/IP-Protokollstapel und ISO/OSI-Modell (Modul 3.14)
- Kommunikationsprotokolle (insbes. TCP und IP, IPv6) (Modul 3.14)
- Routingalgorithmen (Modul 3.14)
- Einfache und zusammengesetzte Abfragen in SQL (Modul 3.13)
- Software-Architekturen inkl. Web-Architekturen (Modul 3.12)
- Infrastruktur (Dateisysteme) (Modul 2.14)
- Assembler, Caches, virtueller Speicher, Stack (Modul 2.13)
- Ist-Analyse, Anforderunganalyse (Modul 2.11)
- bash, html (Module 1.14/1.15)
- Kryptographie (Modul 1.12)

1.1 Zeitplanung

Das wichtigste ist vermutlich zuerst die Zeitplanung: Wann und Wo müssen Sie sich einfinden um an der Veranstaltung teilzunehmen? Wann müssen Sie Ergebnisse abliefern, und wie müssen Sie das? Wann bekommen Sie welches Material und die Bewertung Ihrer Leistung?

Übung	Dienstag	08:00 - 13:00
Vorlesung[1] ab	Donnerstag	08:00
Abgabe Labortagebuch	Freitags	16:00

Arbeitsbelastung Selbstlernzeit

(4:40 h/Woche)

Entwurf:	2-3 h/Woche
Labortagebug:	0,25 h/Woche (Mittelwert)
Übungsvorbereitung: 2-3 h/Woche	

Arbeitsbelastung Präsenzzeit

(1:30 h/Woche)

Vorlesungsvideos:	1 5h/Woche
Präsenzübung:	1:50 h/Woche

Präsenzzeit: 2 VL, 2 Ü (3h 20m)

Selbstlernzeit: (4h 40m)

Semesterzeiten: 14 Wochen

- Vorlesungsstart: 20.10.2020
- Prüfungsanmeldung: bis Ende 3. Woche
- Vorlesungsende: 04.02.2021

Zielgruppe:

Studiengänge: INF und WINF

- Fachsemester: 5 (Pflichtveranstaltung)

Prüfungsform: Entwurf (siehe unten)

1.2 Arbeitsorganisation

1.2.1 Lern- und Prüfungsphasen

Es gibt einen grundsätzlichen Unterschied zwischen der Zeit vor und der Zeit nach dem Start einer Prüfungsleistung. Grundsätzlich gilt:

> Probleme müssen rechtzeitig **vor** dem Start- bzw. Abgabetermin einer einzelnen Prüfungsleistung mitgeteilt werden.

Mit dem Start (beziehungsweise der Abgabe) einer Prüfungsleistung ändert sich der Arbeitsmodus von “Lernen und Lehren” zu “Prüfen”. Im Modus “Lernen und Lehren” möchte ich Ihnen jede Unterstützung angedeihen lassen die möglich ist um ihren Lernerfolg zu fördern. Im Modus “Prüfen” ist es meine Aufgabe diesen Lernerfolg zu prüfen und zu bewerten und damit auch den Wert eines erfolgreichen Kursabschlusses sicherzustellen. Praktisch bedeutet dies, dass es nach dem Start einer Prüfungsleistung (beziehungsweise dem Abgabezeitpunkt von Übungsaufgaben) keine Entgegenkommen bezüglich Terminen, Prüfungsinhalten oder sonstige Anpassungen der Lehre und Prüfung geben kann. Natürlich stehen ihnen alle vorgesehenen formalen Wege, wie die Einreichung von ärztlichen Attesten oder die Anrufung des Prüfungsausschusses offen.

Im Anschluß an die Prüfungsphase haben Sie aber immer die Gelegenheit die Bewertung ihrer Prüfungsleistung einzusehen und die Korrektur von Bewertungsfehlern einzufordern.

Abbildung 1.1: Lern-, Lehr- und Prüfungsphasen

1.2.2 Gruppen

Wir unterscheiden in diesem Kurs Arbeitsgruppen und Präsenzgruppen.

Mit Präsenzgruppen bezeichnen wir die Einteilung in die Übungsgruppen. Die Einteilung in dies Übungsgruppen werde ich über Elli einrichten.

In den Arbeitsgruppen bearbeiten Sie Übungs- und Semesteraufgabe. Arbeitsgruppen bestehen idealerweise aus drei Studierenden. (Ohne Corona würden wir die Arbeitsgruppen für die Übungen regelmäßig neu würfeln, dieses Semester lassen wir das mal.) Die Arbeitsgruppen finden sich in der ersten Übung zusammen. Sie dürfen sich da gerne Personen suchen mit denen Sie gut zusammenarbeiten können.

Kurz:

- Arbeitsgruppe
  - Übungen und Semesterentwurf
  - 3 Studierende
  - Einteilung automatisch
- Präsenzgruppen (Mittwochs Z 2360)
  - Gruppe 1: 08:00 - 09:30
  - Gruppe 2: 09:45 - 11:15
  - Gruppe 3: 11:30 - 13:00

1.3 Lern- und Lehrkonzept

Dieser Kurs lehnt sich an das Konzept des *Inverted Classroom* an. Dabei werden die Vorlesungen als Videos bereitgestellt und sind von den Studierenden als Vorbereitung der Präsenzveranstaltungen zuhause durchzuarbeiten. Die

Präsenzveranstaltungen können dann stärker für interaktive Lehrformate genutzt werden. In diesem Kurs werden die Präsenzveranzveranstaltungen einmal mit dem ganzen Kurs als Plenum und in Gruppenarbeit als Übung durchgeführt. Dabei sollen wesentliche Teile der bisherigen Vertiefung im Selbststudium (umgangssprachlich “Hausaufgaben”) in den Übungen durchgeführt werden.

1.3.1 Vorlesung (Video)

Die Vorlesung wird in der Form von Kurzvideos als Selbstlernteil, nach eigener Zeitvorgabe, durchgeführt. Die Videos werden jeweils durch leichte Verständnisfragen abgeschlossen. Die Beantwortung dieser Fragen ist nicht Teil der Prüfungsleistung. Die Fragen müssen aber korrekt beantwortet werden, bevor das nächste Video angeschaut wird.

Die Vorlesung ist in thematische Unterabschnitte unterteilt. Die Themen bauen teilweise aufeinander auf. Inhalte der Unterabschnitte werden in den Übungsaufgaben vertieft und müssen dementsprechend zum Abgabezeitpunkt der Übungsaufgaben erfolgreich abgeschlossen sein.

1.3.2 Präsenzübung

In der Übung wird anhand von konkreten Aufgabenstellungen der Inhalt der vorangegangenen Vorlesung erarbeitet und erweitert. Die Aufgaben sind derart gestaltet, dass in der Übungsstunde die von den Studierenden vorbereitete, Lösungen diskutiert werden.

Musterlösungen werden in der Regel nicht vom Lehrstuhl ausgegeben, es besteht die Möglichkeit, dass Studierende ihre eigenen Musterlösungen in den Ellie-Foren, nach der Abgabefrist veröffentlichen und diskutieren.

Die Übungen finden jeweils in Präsenzgruppen (ca. 20 Personen) statt. In den Übungen arbeiten die Studierenden in betreuten Kleingruppen an den Übungsaufgaben. In den Präsenzübungen finden weiterhin regelmäßige Konsultationen zur Semesteraufgabe statt.

1.3.3 Entwurf

Über das Semester wird von jeder Gruppe eine Entwurfsarbeit als Semesteraufgabe angefertigt. Die Themen werden in der zweiten Semesterwoche vergeben.

- Dokumentierte Sicherheitsanalyse
- Sicherheitsinfrastruktur

  - Threat Intelligence Server (MISP) mit Integration von Hackmageddon
  - Snort installation
  - Host-IDS mit Tripwire
  - Network logging mit Netflow
  - Sicheres Terminal Session Sharing (tmux)
  - Konzept VPN für Vorlesungen
  - RBAC Schutzmodell in SELinux für Hopper
  - WebID für die Hochschule
  - Email-Verification RA bauen
  - Sicheres Konzept SMTP-Relay
  - Analyse der Exploit Trader
  - (weitere Sicherheitsanwendungen nach Absprache)
- Demonstratoren
  - Spectre/Meltdown Simulation
  - Webserver/Webservices
  - NAT Slipstreaming Demo
  - ...
- Umfang (10 Seiten ohne Anhang und Literaturliste)
- Struktur
  - Zusammenfassung (1 Absatz)
  - Einleitung (1 Seite)
    * Problemstellung
    * Einordnung in IT-Sicherheit
    * Lösungsidee
    * Vorgehen
    * Struktur des Dokumentes
  - Technische Grundlagen (ca. 1 Seite)
    * Hintergrundwissen (nur notwendiges)
  - Hauptabschnitte (6 Seiten)

        * Konzept (e. g., 3 Seiten)
        * Implementation (e. g., 1 Seite)
        * Dokumentation (e. g., 2 Seiten)
    - Fazit (max. 1 Seite)
        * Ergebniszusammenfassung
        * Bewertung der Ergebnisse
        * Future Work/Ausblick

Weitere Themen nach Absprache.

1.4 Kursunterlagen

Vorlesungsvideos

Übungsaufgaben

Vertiefende Quellen

Skript: für ausgewählte Teile der Vorlesung wird zusätzlich ein begleitendes Skript bereitgestellt. Das Skript ist liegt größtenteils in englischer Sprache vor und enthält eine Obermenge der Inhalte des Kurses. Das Studium des Skriptes ist kein vollwertiger Ersatz der konstanten Teilnahme am Kurs.

1.4.1 Forum und Chat

Ankündigungen und Kursinformationen werde ich über die Matrixgruppe ws21-itsec bekanntgeben. Für die inhaltliche Diskussion zum Thema IT-Sicherheit steht ausserdem die Gruppe IT-Sec Meetup zur Verfügung.

1.5 Prüfungsbedingungen

Die Prüfung besteht in diesem Semester aus einer Entwurfsarbeit, welche in der Arbeitsgruppe als Gemeinschaftsleistung abzuleisten ist. Die Prüfungsleistung besteht dabei aus einer Ausarbeitung mit Anhang und dem Quellcode eines Prototypen.

Semesteraufgabe Gruppenleistung

- Systementwurf
- Inhalt nach Absprache

- Abgabe zum Vorlesungsende
- Umfang: 2h Wochenarbeitszeit, 5-10 Seiten Bericht (inkl. *Code*)
- Dringend beachten: Tutorial Howto Schreiben

Labortagebuch
- Wöchentliche Abgabe
- Kryptographisch signiert

basierend auf Übungsaufgaben und Protokollen

Autor:in Name und Emailadresse

Datum RFC 3339

Ablageort `hopper:/home/itsec/2021/labortagebuch/`

Dateiname `<Datum>-<uname>.log`, e. g., `2021-10-22-lafischer.log`

Dateiformat S/MIME signiert `openssl smime -sign -signer <cert> -inkey <key>`

Logeinträge
- Semesterprojekt
- Vorlesung
- Übung

Arbeitszeit angeben

1.5.1 Übungsaufgaben

Die Übungsaufgaben ergänzen und erweitern die Vorlesungsinhalte und sollen wesentlich in der Präsenzübung behandelt werden. Es wird erwartet, dass die Studierenden sich die nötigen Arbeitsmittel (Bücher, Anleitungen,...) vor der Präsenzübung einsatzbereit vorbereitet haben.

Die Übungsaufgaben werden nicht bewertet, die Dokumentation der Bearbeitung ist wesentlicher Teil der Labortagebücher.

1.5.2 Semesteraufgabe

Die Semesteraufgabe soll als Gruppenaufgabe eine Klammer über das gesamte Semester bieten und eine Möglichkeit zur punktuellen Vertiefung bieten. Das Thema der Aufgabe soll jeweils die praktische Umsetzung einer Sicherheitsinfrastruktur inklusive einer Dokumentation für Maintainer und Nutzer darstellen.

Ein Beispiel wäre die Einrichtung und Parametrierung einer Firewall. Die Dokumentation umfasst dabei, neben den Filterregeln, eine anforderungsbasierte Produktauswahl, Dokumentation und Begründung der Auswahl des Regelwerkes, Anleitung für die Wartung und eine Beschreibung der Filtersemantik für betroffene Endnutzer (z.B. Mitarbeiter der Firma)

```
Author: lafischer <lafischer@hs-bremerhaven.de>
Date:   2021-10-22T15:37:13+02:00

Semesterprojekt:
- Entwurf für Zero-Knowledge Authentication aufgeschrieben (1.Entwurf)
- Issue 13: IMAP fetch mittels eigenem Bash-Skript gelöst
  + Lösung in Projektsitzung vorgestellt
- Zeit: 3:20h

Vorlesung:
- TLS: Kann man TLS auch für asynchrone Verbindungen nutzen?
  + RFC 8446 durchgearbeitet, aber keine Lösung gefunden
- Vorlesungsskript durchgearbeitet
  + Seite 17, Satz 3 fehlt ein Komma
- Zeit: 1:10h

Übung:
- X509 Zertifikat in Server konfiguriert
- Erfolgreicher Verbindungsaufbau mit Browser
- Zeit: 0:30h
```

Abbildung 1.2: Beispiel für einen Logbucheintrag

Die Dokumentation der Semesteraufgabe soll vom Umfang her 10 bis 15 Seiten Inhalt (ohne Titelseite, Referenzen und Anhang), oder fünf Seiten pro Mitglied der Arbeitsgruppe, nicht überschreiten. Die Abgabe erfolgt als PDF mit LaTeX-Quelltext. Die Ausarbeitung muss im bereitgestellten Template [2] abgefasst werden. Der Abgabeort wird rechtzeitig über die offiziellen Kanäle kommuniziert.

1.6 Quellen

Die Vorlesung basiert grundlegend auf den Lehrbüchern von William Stallings: „Computer Security“ [**stallings2008computer**], „Cryptography and Network Security“[**Stallings11Cryptograph**
und „Network Security Essentials“ [**Stallings07NetworkSecurityEssentials**],
sowie dem Buch „IT-Sicherheit: Konzepte - Verfahren - Protokolle“ von Prof. Dr. Claudia Eckert [5]. Diese sind entsprechend mit Seitenzahlen referenziert und verweisen in der Regel auf weitere Quellen.

Es wird erwartet, dass Teilnehmer dieser Veranstaltung zuerst selbstständig diese Werke konsultieren.

- Willam Stallings:

[2] `https://informatik.hs-bremerhaven.de/lafischer/lehre/leistungsnachweis.zip`

```
% Semesteraufgabe IT Sicherheit Titel
% Namen der Beteiligten (Gruppe XX)
% Wintersemester 2020

Abstract (max 400 Zeichen)

Libero. Feugiat a, suspendisse per, primis sociis. Nascetur, elementur
hymenaeos curae, proin dis ipsum eros potenti. Porta. Purus. Vel
senectus fusce elit vitae, lorem, a, lacinia litora, hac. Vitae cum
duis. Ac litora ac sed venenatis congue parturient netus. Mattis nunc
eros id porta quis feugiat eu, libero. Proin amet nisl tempus, incepto
Ullamcorper ad, aenean. Consectetuer dapibus justo mus enim nisl vive
est.


## Ziel

## Umsetzung

## Evaluation
```

Abbildung 1.3: Semesteraufgabe Template

  - „Computer Security“
  - „Cryptography and Network Security“
  - „Network Security Essentials“
- Claudia Eckert: „IT-Sicherheit: Konzepte - Verfahren - Protokolle“

Praktische Anteile der Vorlesung finden im Lern- und Lehrlabor IT-Sicherheit statt. Der Zugang zum Labor findet über das Virtual Private Network (VPN) `wireguard` statt. Damit Sie an der Veranstaltung teilnehmen können, müssen Sie deshalb die nötige Software und Schlüssel installieren.

1. Installieren Sie wireguard entsprechend der Anleitung
2. Der Zugangspunkt wird über die Webseite bekanntgegeben
3. In der Übung werden Sie angeleitet den Schlüssel abzulegen
4. Ihr Zugang wird dann freigeschaltet

1.7 Welcome to Security

Informatik liefert Werkzeuge für nahezu alle Geschäftsbereiche und Technologien. Schon heutzutage gibt es fast keinen Lebensbereich der nicht von Computertechnologie durchdrungen ist.

Sichere...

- Infrastruktur Aufbau und Betrieb
- Software-Engineering
- Kommunikation
- Entwurf
- Hardware
- ...

IT-Sicherheit nutzt, auf der anderen Seite, eine Vielzahl unterschiedlicher Grundlagen aus verschiedenen Disziplinen.

- Zahlentheorie & Algebra
- Graphentheorie
- Stochastik
- Kommunikationsnetze
- Management (Organisation/Prozesse)

Das bedeutet, für diesen Kurs, dass wir das Vorwissen aus eigentlich allen bisherigen Kursen und Vorlesungen an verschiedenen Stellen benötigen werden.

1.8 Game of Keys

Ziel des Spieles ist es den Umgang mit Zertifikaten und Schlüsseln nachhaltig zu lernen. Dafür verwenden wir OpenSSL, eine weit verbreitete Bibliothek, die in verschiedensten Servern und Dienste verwendet wird. Wir werden den Umgang grundsätzlich auf der Kommandozeile lernen und dabei die verschiedenen Funktionen während des ganzen Semesters nutzen. Praktisch bedeutet dies, dass es in diesem Kurs keine Kommunikation geben wird, die nicht abgesichert ist.

- Alle Kommunikation ist Sender-authentifiziert
  - Laborlogbuch (wöchentlich)

  - Emailanfragen
  - Entwurfsabgabe
- Certification Authority
  - `hopper:/home/sireal-ca`
  - `sirealca@hs-bremerhaven.de`
  -

Digital certificates are omnipresent. They are needed for secure communication or to prove access rights and qualifications. But, what exactly is a “digital certificate” in IT-Security?

A certificate

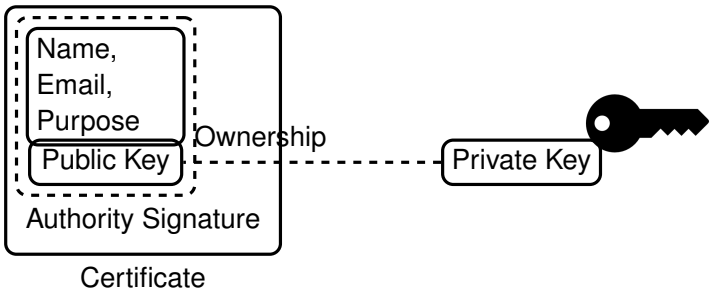
- relates list of **attributes**
- authenticity of **provable authority**
- allows proof of **ownership**

Example: Passport

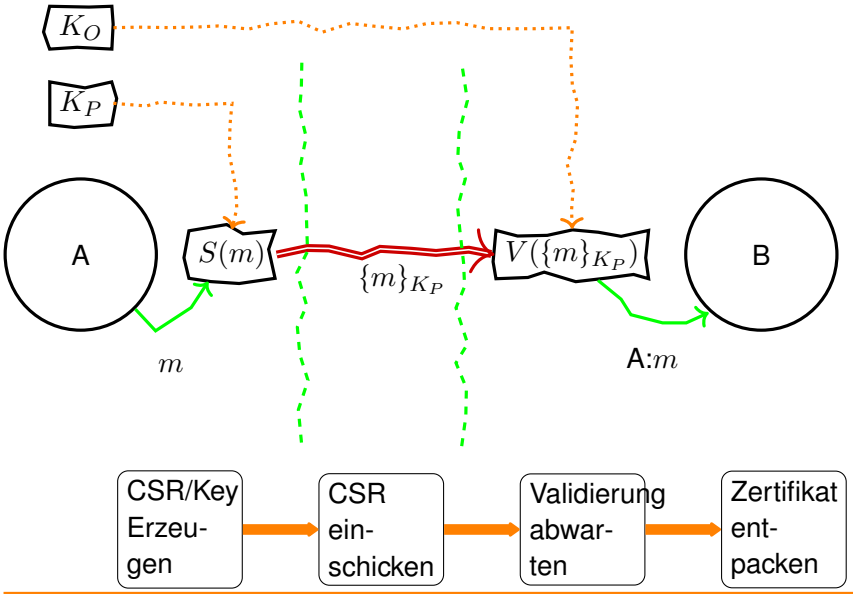
Attributes
- Name,
- Birthday,
- Place of Birth,
- Unique ID,...

Provable Authority
- Security Features,
- official seal,
- well-known design

Ownership
- Biometric data,
- physical ownership



Asymmetric cryptography provides a completely new application for cryptographic algorithms as compared to symmetric cryptography. Using a private key is an operation only the holder of that key can facilitate. This allows, for example, to interpret the result of that operation as a proof that the holder of that private key has seen the message and approves it. The most common application of this is to provide authentication of the source of a message, or as it is commonly refered to, a *signature*.



Zeilen für Erzeugung eines Requests fehlen

- sec-user.conf
- Passwort für Revocation benötigt. Merken!
- CSR darf(!) nicht öffentlich lesbar sein

Konfiguration siehe Veranstaltungsseite.

- CA-Büro: `hopper:/home/sireal-ca`
- Ablageort: CA-Büro `cert-requests`
  - Dateiname: `<local email>-<serial>.csr`
  - Dateirechte: `g+rw`
  - Eigentümer: `<user>:sireal-ca`
- Abholung in: CA-Büro `certs`

Zertifikate sind öffentlich.

S/MIME Signatur

Zeilen für die Erzeugung der Signatur fehlen

Punktgewinn:

- Signieren einer eigenen Einreichung mit dem Schlüssel einer anderen Person, e. g., :
  - Auslesen des privaten Schlüssels
  - Erschleichen eines neuen Zertifikats für die andere Person
- Erfolgreichen Angriff durchführen
- Einreichung eines Proof-of-Concept

  (PoC) per Email lars.fischer
- Im Entwurfsanhang aufführen

Punktabzug:

- Ursache: Kompromittierter Schlüssel
- Symptom: Gestohlene Identität
- Abhilfe: Zertifikat erneuern
- Email an sirealca
- Absender: wie im Zertifikat
- Subject: “revoke”
- Angabe des Challenge-Passwortes und alter Seriennummer
- Ablage eines neuen CSR mit inkrementierter Seriennummer

1

Introduction

Vorstellung

Contents

1.0.1 Objectives of this course

Basic Security Knowledge:

- Security as a Process
- Threats, Vulnerabilities and Controls
- Communication and Protocols
- Architectures
- Computer Systems
- Software Security

1.1 IT-Security Objectives

The central question of *security* is one of authorisation. That is whether a specific *action* by a defined *subject* is allowed at the given circumstances to be executed on a given *object*. Work on the topic security is concerned expression and definition of allowed vs. forbidden, prevention and detection of misbehaviour, and the recovery from incidents.

> **Computer Security:** The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications) [NIST Computer Security Handbook, 1995]

1.1.1 CIA-Triad

The central concept of security is that of *Authorization*. With respect to authorization, IT-Security is understood to be defined by classes security-objectives. The most common understanding is given by the “CIA-Triad” in [6]:

Confidentiality *(Geheimhaltung)*Only authorized users get the information.

Integrity *(Integrität)* Information is correct, complete, and current or this is detectably not the case.

Availability *(Verfügbarkeit)* Information and resources are accessible where and when the authorized user requires them.

e. g., [NIST Computer Security Handbook, 1995]

Although there are different opinions in the community whether this classisfication is too granular or completely covering all nuances of security, but the CIA-Triad provides an accepted baseline for the understanding of the scope of IT-Security. The terminology is applied in a very practical manner, with subcategories and limited extensions if the need arises.

And, while the dicipline focusses on information technology it must be stressed, that it can not be expected that attackers limit themselves only to digital attack vectors. Defenders can only be successful, if they also don't limit their methods to the digital domain but integrate security into the larger environment where IT is embedded in or executing control.

The common ground is the understanding, that security is the topic of intentional unauthorised actions by threat actors and protections and controls by defenders or asset-owners.

1.1.2 Further Security Related Objectives

The CIA-Triad 1.1.1 provides a categorisation requires some refinement into sub-objectives to allow work on solutions. There is no general consensus about refined sub-objectives, but a widely used of common terms that are to be introduced here.

In Germany the canonical reference on IT-Security by Claudia Eckert distinguishes six different categories of security objectives. Figure 1.1 provides a mapping of objectives.

Authenticity *(Authentizität)* denotes that the source of an information object can be proven. This implies that the integrity of that object is also protected. Therefore this is considered a part of the Integrity category in the CIA-Triad. (See 1.1) It can be argued that the term *integrity* is not sufficiently covering specific problems of authentication.

Integrity *(Integrität)* has the same meaning than the same term in the CIA-Triad without authentication.

Confidentiality *(Vertraulichkeit)* similar to the CIA-Triad, although in this definition it is not addressing specific topics from the domain of privacy protection.

Availability *(Verfügbarkeit)* denotes roughly the same topics as in the CIA-Triad.

Non-Repudiation *(Unabstreitbarkeit)* denotes the specific problems related to proving that defined actions have taken place, i. e., to proof that some other party has signed a contract. But also problems of secure logging of events where the other party is not actively providing proof of its actions. Techniques and problems are often similar to the topices of the Integrity-category in the CIA-Triad, but this category also often has requirements related to availability, e. g., availability of a current and correct time.

Ano-/Pseudonymity *(Anonymität/Pseuyonymität)* This term covers specific problems in privacy protection. The techniques and proceedures differ from the cryptographic techniques used for protection of confidentiality (in the Eckert-Model). One requirement, for example, could be unobservability of communication. It might be obvious that Anonymity-protection is in direct conflict with Authentication.

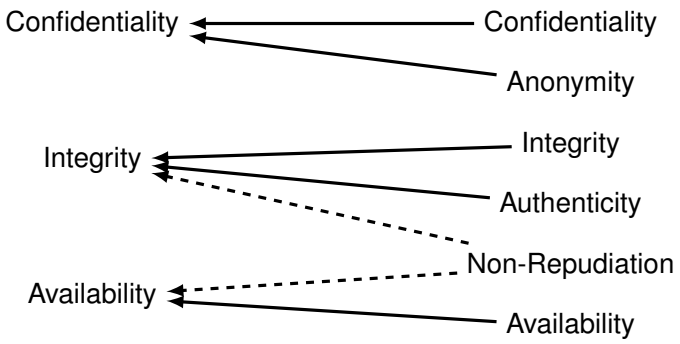


Figure 1.1: Mapping Eckert as sub-objectives onto the CIA-Triad.

1.1.3 Interedependencies of Security Objectives

> Security is a process, not a product.
>
> Products provide some protection, but the only way to effectively do business in an insecure world is to put processes in place that recognize the inherent insecurity in the products. The trick is to reduce your risk of exposure regardless of the products or patches.
>
> Bruce Schneier:
> Computer Security: Will We Ever Learn? [seen 2020-08-26]

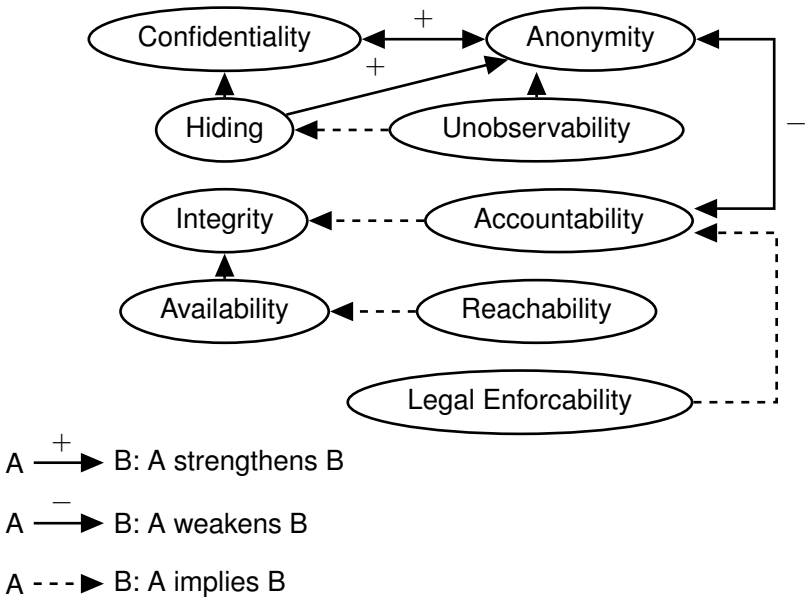


Figure 1.2: The security model of Wolf and Pfitzmann [10]

Aufgabe Sicherheitsanforderungen Wahlen/Onlin Wahlen, Hausaufgabe Recherche, PÜbung Formulierung eines Sicherheitsmodels

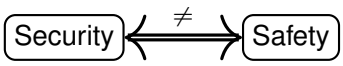
1.2 Security vs. Safety

Dieser Abschnitt basiert auf einem Artikel von Fischer und Messerschmidt für die AG KRITIS. [**FM2020˙safety˙security**]

Verständnisfragen Begriffe

Wo der Brandschutz einen lebensrettenden Notausgang sieht (*Safety*), sieht der Sicherheitsberater eine Schwachstelle im Zugangsschutz (*Security*). Dieses simple Beispiel zeigt den Konflikt, der oft zwischen dem besteht, was im Englischen als *Security* und *Safety* unterschieden wird. Im Kern des Konflikts steht die Frage, welche Sicherheit für welche Schutzfunktionen benötigt werden und welche Schutzmaßnahmen notwendige Sicherheitsmaßnahmen untergraben. Die Beschäftigung mit dieser Frage ist notwendig, weil die beiden Seiten oftmals von unterschiedlichen Experten bearbeitet werden. Kommunikation ist notwendig, um die unterschiedlichen Ziele miteinander zu verbinden und Missverständnisse zu vermeiden.

Kommunikation setzt ein gemeinsames Verständnis der Begrifflichkeiten voraus, weshalb wir im Folgenden den Versuch einer Begriffsbestimmung der *Sicherheit* durch die zwei Begriffe *Safety* und *Security* unternehmen wollen. In der Literatur finden sich verschiedene Merkmale die zur Unterscheidung herangezogen werden.



„Sicherheit" ≠ „Sicherheit"

Die Unterscheidung in Kurzform:

1. Unterscheidung nach

   Schadensursache:

   Safety als Schutz eines Systems vor zufälligen, nicht-bewusst herbeigeführten Schadereignissen, z.B. Wetterphänomene oder Fehlbedienung.

   Security als Schutz eines Systems vor bewußt herbeigeführten, zielgerichteten Schadereignissen, z.B. Cyberangriffe.

2. Unterscheidung nach Art des Schadens und nach Schädigungsrichtung:

   Safety als Schutz vor Personenschäden, in der Regel durch das betrachtete System.

   Security als Schutz vor Schaden am betrachteten System.

Diese Situation zeigt dass auch grundlegende Begriffe komplexer sind als vielfach angenommen. Das ist natürlich unbefriedigend und erfordert, dass das Verständnis der Sicherheitsgrundbegriffe neu ausgehandelt werden muss.

Safety und Security nach dem Ursachekriterium

Von *Safety* sprechen wir, wenn es darum geht Anlagen, Prozesse oder Personen vor Beeinträchtigung durch ungesteuerte, zufällige oder natürliche Ereignisse zu schützen. In diese Klasse fallen Ereignisse die durch "ungewollte Fehlbedienung" ausgelöst werden, ebenso wie die Beschädigung durch Umweltereignisse wie Erdbeben oder Stürme.

Demgegenüber verstehen wir *Security* als Verhinderung oder Vermeidung von unerlaubter, absichtlicher Beeinflussung, mit dem Willen zur Schädigung von *Werten* (*Assets*). Im Gegensatz zur Safety sind der wesentliche Faktor im Bereich Security die *Angreifer* (*Attacker* oder auch *Threat Agents*), die sich insbesondere dadurch auszeichnen, dass sich ihre Fähigkeiten und ihr Verhalten schlecht vorhersagen lassen. Während es vergleichsweise einfach ist, die erwarteten Sturmereignisse in einer Region historisch aufzuzeichnen, statistisch zu beschreiben und damit zu prognostizieren, ist das Verhalten von Angreifern nicht durch empirische Beobachtung vorhersagbar. Schlimmer noch, es ist anzunehmen, dass, wenn Sicherheitsmaßnahmen auf bekannte Angriffsmuster optimiert werden, sich die Angreifer anpassen und insbesondere die verbleibenden Lücken oder genau die Sicherheitsmaßnahmen selbst angreifen.

Die Definition über die Schadensursache ist für die Planung von Schutzmaßnahmen einfach anzuwenden. Bei der Analyse und Reaktion auf Schadensereignisse ist jedoch gerade dieses einfache Kriterium der bewussten Schädigung schwer zu ermitteln. Denn es erfordert eine Attribution der Schadensursache bzw. der Verursacher. Eine eindeutige Attribution ist in einer digitalen Welt aber im Allgemeinen nicht möglich. Deshalb ist es sinnvoll, zusätzlich die Definition nach anderen Kritierien zu berücksichtigen.

Safety und *Security* nach dem Schadenskriterium

Das Kriterium der Schadensrichtung besagt, dass die Unterscheidung sich dadurch ergibt, ob ein möglicher oder tatsächlicher Schaden am betrachteten System oder ein Schaden durch das betrachtete System an einer anderen Sache vorliegt. Der Begriff *Safety* bezieht sich deshalb auf Schaden, der durch das System selbst entsteht, z.B. ein Zug dessen Bremssystem versagt. Der Begriff *Security* bezieht sich auf Schadenswirkung am System.

Das informelle Glossar der Internet Enginieering Task Force, welche wesentliche Internet Standards spezifiziert hat, empfiehlt die folgende Definition:

> **Safety**: “The property of a system being free from risk of causing harm (especially physical harm) to its system entities.” [9]

Sicher, im Sinne von *Safety*, ist ein System dann, wenn kein Risiko besteht, dass von einem betrachteten System Schaden ausgeht.

Für den Begriff *Security* werden, je nach Kontext drei unterschiedliche Definitionen angeboten. Als Zustand wird *Security* als Ergebnis der Einführung und Aufrechterhaltung von Maßnahmen zum Schutz des Systems verstanden. *Security* kann weiterhin als Oberbegriff für genau diese Maßnahmen verstanden werden. Als Gegenstück zur *Safety* wird *Security* analog zur IT-Sicherheit als Zustand in dem ein System frei ist von möglichem Schaden von aussen, in Bezug auf die Unmöglichkeit von unauthorisiertem Zugang, Veränderung, oder Datenverlust, aber auch zufälligen Schadenseinwirkungen.

> **Security**: “A system condition in which system resources are free from unauthorized access and from unauthorized or accidental change, destruction, or loss.” [9]

Safety und *Security* unterscheiden sich auch nach der Art des Schadens. *Safety* wird immer als ein Schutz vor Personenschäden verstanden, also Verletzung oder Tod von Menschen.

Demgegenüber bezieht sich *Security* sowohl auf den Schutz vor Personenschäden wie auch vor materiellen und ideellen Schäden. Deshalb ist es sinnvoll, bei *Security* zusätzlich nach *IT Security* und *Physical Security* zu unterscheiden, um besser nach Schadensart differenzieren zu können.

Der Begriff *IT-Sicherheit* bzw. *IT Security* oder *Computer Security* im Englischen kann als Spezialfall der oben definierten *Security* betrachtet werden. Informationstechnik (IT) ist ein wesentliches Element der aktuellen Veränderungen in Kritischen Infrastrukturen. IT-Sicherheit geht üblicherweise von der Definition des National Institutes for Standards and Technology (NIST) aus. Das NIST definiert *Computer Security* als die Sicherstellung von Integrität, Geheimhaltung und Verfügbarkeit von Systemen und Daten:

> **Computer Security:** The protection afforded to an automated information system in order to attain the applicable objectives of preserving the *integrity*, *availability*, and *confidentiality* of information system resources (includes hardware, software, firmware, information/data, and telecommunications). [**NISTSP800-12**]

Der Begriff *Physical Security* kommt aus dem Militär und bezeichnet im engeren Sinne alle physischen Schutzmaßnahmen gegen unerlaubten Zugriff. Ein vergleichbarer deutscher Begriff ist Objektschutz. Im weiteren Sinne umfasst *Physical Security* alle Schutzmaßnahmen durch Sicherheitskräfte, auch im nicht-militärischen Bereich wie zum Beispiel durch die Polizei.

Während die *IT Security* sich überwiegend auf finanzielle *Werte* bzw. Schäden konzentriert, ist die *Physical Security* mit finanziellen (Raub), menschlichen (Mord, Verletzung, Entführung, Stalking), ideellen (Rufschädigung, Beleidigung), wirtschaftlichen (Sabotage) und gesellschaftlichen (Terrorismus) Schäden bzw. *Werten* befasst.

Das Kriterium der Schadensrichtung besagt, dass die Unterscheidung sich dadurch ergibt, ob ein möglicher oder tatsächlicher Schaden am betrachteten System oder ein Schaden durch das betrachtete System an einer anderen Sache vorliegt. Der Begriff *Safety* bezieht sich deshalb auf Schaden, der durch das System selbst entsteht, z.B. ein Zug dessen Bremssystem versagt. Der Begriff *Security* bezieht sich auf Schadenswirkung am System.

Das informelle Glossar der Internet Enginieering Task Force, welche wesentliche Internet Standards spezifiziert hat, empfiehlt die folgende Definition:

> **Safety**: “The property of a system being free from risk of causing harm (especially physical harm) to its system entities.” [9]

Sicher, im Sinne von *Safety*, ist ein System dann, wenn kein Risiko besteht, dass von einem betrachteten System Schaden ausgeht.

Von *Safety* sprechen wir, wenn es darum geht Anlagen, Prozesse oder Personen vor Beeinträchtigung durch ungesteuerte, zufällige oder natürliche Ereignisse zu schützen. In diese Klasse fallen Ereignisse die durch “ungewollte Fehlbedienung” ausgelöst werden, ebenso wie die Beschädigung durch Umweltereignisse wie Erdbeben oder Stürme.

Demgegenüber verstehen wir *Security* als Verhinderung oder Vermeidung von unerlaubter, absichtlicher Beeinflussung, mit dem Willen zur Schädigung von *Werten* (*Assets*). Im Gegensatz zur Safety sind der wesentliche Faktor im Bereich Security die *Angreifer* (*Attacker* oder auch *Threat Agents*), die sich insbesondere dadurch auszeichnen, dass sich ihre Fähigkeiten und ihr Verhalten schlecht vorhersagen lassen. Während es vergleichsweise einfach ist, die erwarteten Sturmereignisse in einer Region historisch aufzuzeichnen, statistisch zu beschreiben und damit zu prognostizieren, ist das Verhalten von Angreifern nicht durch empirische Beobachtung vorhersagbar. Schlimmer noch, es ist anzunehmen, dass, wenn Sicherheitsmaßnahmen auf bekannte Angriffsmuster optimiert werden, sich die Angreifer anpassen und insbesondere die verbleibenden Lücken oder genau die Sicherheitsmaßnahmen selbst angreifen.

Die Definition über die Schadensursache ist für die Planung von Schutzmaßnahmen einfach anzuwenden. Bei der Analyse und Reaktion auf Schadensereignisse ist jedoch gerade dieses einfache Kriterium der bewussten Schädigung schwer zu ermitteln. Denn es erfordert eine Attribution der Schadensursache bzw. der Verursacher. Eine eindeutige Attribution ist in einer digitalen Welt aber im Allgemeinen nicht möglich. Deshalb ist es sinnvoll, zusätzlich die Definition nach anderen Kritierien zu berücksichtigen.

1.2.1 Security Priorities

Selection an prioritisation of security objectives in practice depend obviously on application- and system-specific requirements. But often you will find that security experts prioritise secrecy above integrity with availability being considered sometimes in the aftermath. This is only noteworthy because it is a common point of conflict with other stakeholder groups.

For most cyber-physical systems Safety provides the key objective that comes on top.

One example is conflicting priorities with security requirements in Industrial Control Systems (ICS), or more generally Cyber-Physical System (CPS), shown in Figure 1.3. Physical industrial processes often dependend on timely provision of precursors in the process. Safety-related parts of physical processes even must be functional at all times and under tight stress-conditions. These high requirements on availability in ICS transfer directly to the IT-System that controls these measures.

1.3 Basic Threat Model

The “Internet Security Glossary, Version 2” of 2007 [9] defines attacks graphically as intentional actions of an attacker (threat agent) towards a vulnerability

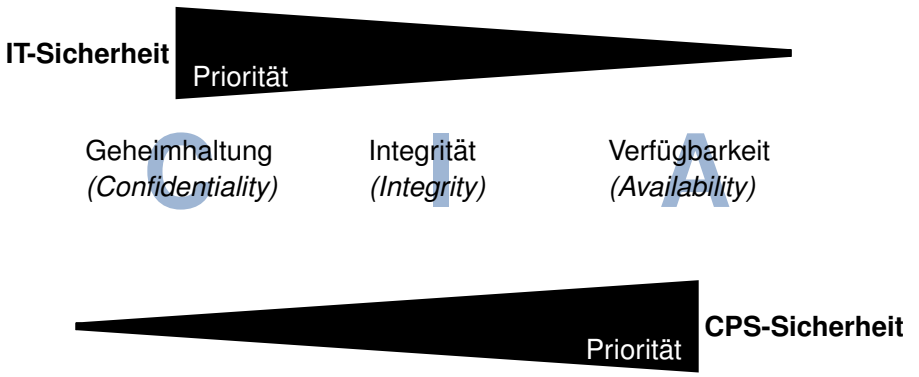


Figure 1.3: Prioritätsumkehr der Sicherheitsziele von information technology (IT)-Sicherheit und CPS-Sicherheit

in an attacked system resource. The interaction is hindered by countermeasures. The model distinguishes attacks (threat actions) as active in the case of of bi-directional interaction and passive where an attacker is only receiving.

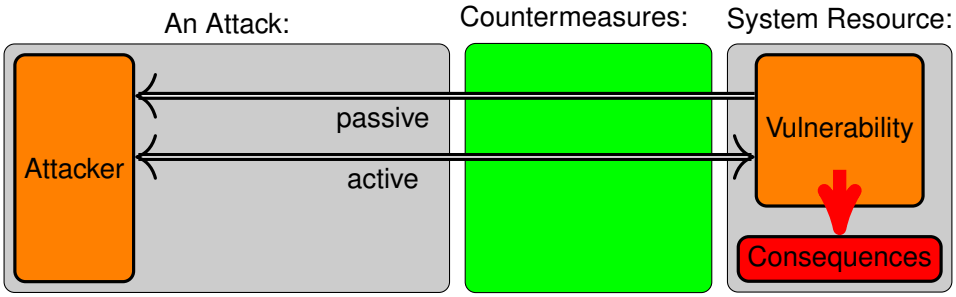


Figure 1.4: Basic model of an attack [9]

Abbildung 12.10 illustriert den Zusammenhang zwischen Angreifern (Threat Agents) und Eigentümern (Owners) von Sachen (Assets). Eigentümern eines Assets Zeichnen sich insbesondere dadurch aus, dass sie, innerhalb eines gegebenen sozialen und rechtlichen Kontext, über die Verwendung eines Asssets Entscheiden dürfen. Das Verhältnis zwischen Angreifern und Eigentümern zeichnet sich wiederum dadurch aus, das sie unterschiedliche Ziele bezüglich der Geheimhaltung, Integrität oder Verfügbarkeit der betreffenden Sachen haben.

Angreifer stellen eine Bedrohung (Threat) dar, wenn sie ihre Ziele mit oder an der Sache, ohne die nötigen Rechte zur Durchführung[1], durchsetzen, oder planen durchzusetzen. Bedrohungen können sich wiederum nur durch die Ausnutzung von Schwachstellen (Vulnerabilities) manifestieren. Die Möglichkeit dass

[1] innerhalb eines verbreitet akzeptierten Kontext von rechtlichen, sozialen und ethischen Richtlinien

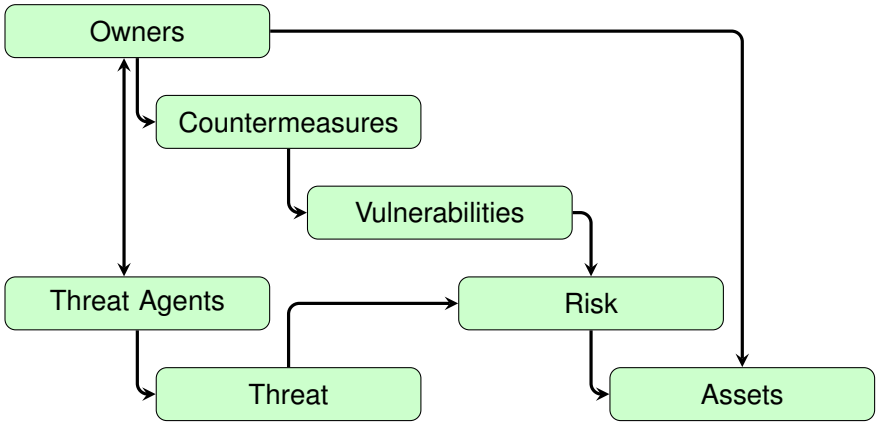


Figure 1.5: Security concepts and relationships [**stallings2008computer**][**CommonCriteria**]

eine Bedrohung sich an einer Sache manifestiert wird als Risiko (Risk) bezeichnet. Um dies zu verhindern, können Eigentümer Maßnahmen (Countermeasures) gegen Schwachstellen ergreifen.

Ein einfaches Beispiel, welches vermutlich jedem aus dem eigenen Leben vorstellbar ist, ist der Einbau eines neuen Schließzylinders in die Wohnungstür um der Bedrohung durch einen neugierigen Vermieter zu begegnen. Dadurch kann die Schwachstelle eines vermuteten Zweitschlüssels im Besitz des neugierigen Vermieters geschlossen werden und das Risiko, dass der neugierige Vermieter sich unbefugt Zugang (das Asset) zur eigenen Wohnung verschafft reduziert werden. Es mag sein, dass die Rechtslage vorsieht, dass der Vermieter ein Recht auf den freien Zugang zur Wohnung hat, dann hätten wir mit unserem Zylindertausch einen Angriff auf dieses Recht durchgeführt.

Interessanterweise kann es sogar so sein, dass dieselbe Handlung in unterschiedlichen Kontexten erlaubt oder verboten ist. Und natürlich ist deshalb auch die Unterscheidung zwischen Angriffswerkzeugen und Werkzeugen nicht immer eindeutig möglich.

Attack A deliberate attempt to assault an asset, not something that is just happening by chance. We can distinguish between passive and active attacks. An active attack tries to alter system resources, while a passive attack tries to “learn”.

Adversary/Attacker/Threat Agent Entity that performs threat actions, i. e., participates in an attack.

Risk $R = D \times P$ An expection of loss, expressing that a given threat may manifest by exploiting vulnerabilities. Usually expresses an expected damage, defined as mean damage D weighted by probability P of an event.

Threat Circumstance, capability, action/event that may lead to exploits of vulnerabilities

Vulnerability A weakness that could be exploited.

Weakness Some point where a system can become vulnerable. As long as this weakness is not directly exposed, it is not actually a vulnerability that can be exploited, but it may become a vulnerability if circumstances expose this weakness to an attacker.

Countermeasure

Attack vector Threat delivery techniques (e-mail) An attack vector is a technique, path or means to deliver a payload or produce a malicious outcome. This may include an exploit, email-attachment or denote more specific activities addressed as some part of a system.

[Internet Security Glossary, Version 2]

1.4 Cyber-Kill Chain

1.4.1 Tactics, Techniques, and Proceedures (TTP)

Tactics, Techniques, and Proceedures (TTP) ist ein Begriff der die üblichen Werkzeuge und Vorgehensweisen einer Gruppe von Angreifern adressiert. Bei der Analyse eines Sicherheitsvorfalls sind die beobachteten TTP häufig ein erster, und auch vergleichsweise stichhaltiger Indikator auf die Herkunft eines Angriffs.

Auf der anderen Seite können Angreifer die TTP einer anderen Gruppe imitieren um die Herkunft eines Angriffs zu verschleiern. Werkzeuge werden genau deshalb oft “veröffentlicht”. Weniger leicht lassen sich allerdings festgefügte Prozesse der Angriffsteams ändern.

1.5 Vulnerabilities

Vulnerability: eine ausnutzbare Schwachstelle

1.5.1 Heartbleed Bug

Heartbleed has been discovered in April 2014 and published as CVE 2014-0160. Reportedly two-thirds of all web-servers have been afflicted by the bug[2]. Active exploitation of this vulnerability has been observed since 2013. The

[2] `https://arstechnica.com/information-technology/2014/04/critical-crypto-bug-in-openssl-opens-two-thirds-of-the-web-to-eavesdropping/`

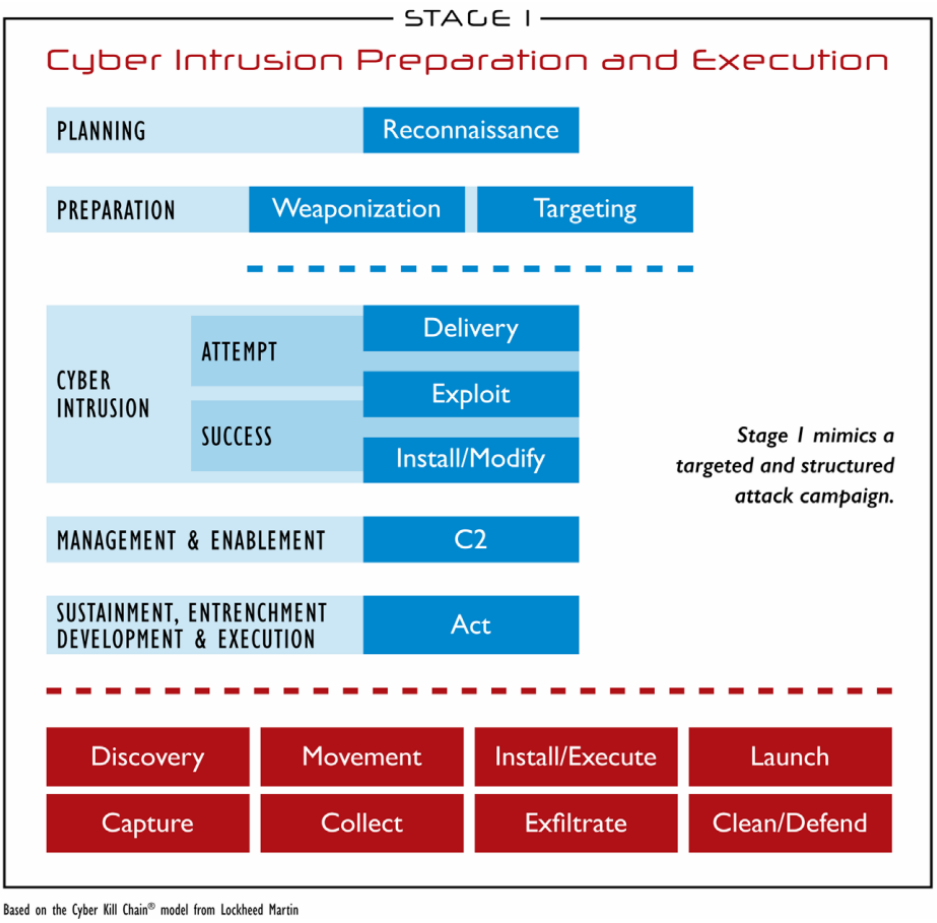


Figure 1.6: Industrial Control System Cyber Kill Chain, first stage representing the classical attack process in IT systems as defined in [7], taken from [1].

Dear SSL-Server, I am a message of 5000 characters. Please reply to me with this exact message.

Here is your message, which contains the 91 characters you sent me and 4909 characters from my process memory right after where I stored your message.

The vulnerable code can be reduced to a single line of code.

Problem:

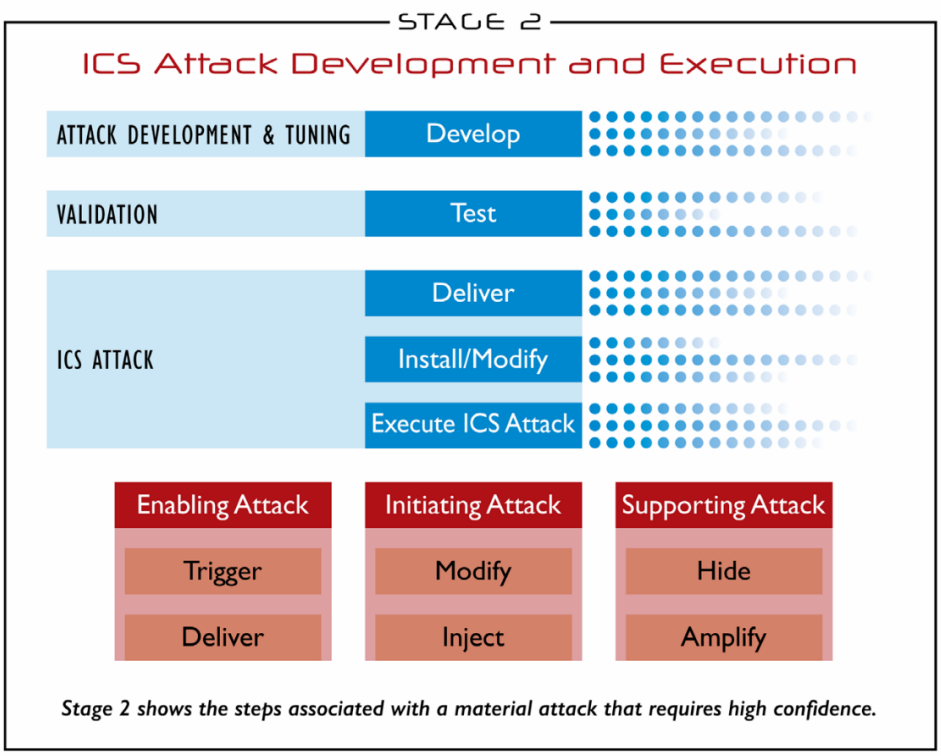


Figure 1.7: Industrial Control System Cyber Kill Chain, second stage describing the attack process into the ICS, taken from [1].

```
  memcpy(bp, pl, payload);
```

Patch:

```
/* Check length != 0 */
if (1 + 2 + 16 > s->s3->relent) return 0;
/* Check payload field fits actual length */
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length) return 0;
/* silently discard per RFC 6520 sec. 4 */
pl = p;
```

The problem that it is never checked whether the length of a received message, stored at `pl` not longer than the length given in `payload`. The value of `payload` is taken at face value from the received message. As a result, an arbitrary number of bytes is copied from `pl` to `bp`. The pointer `bp` points to the return-message that is subsequently sent as response to `pl`.

The vulnerability is exploited by providing a value in `payload` larger than the

length of the message stored at `pl`. That way the server sends additional bytes from the memory of the OpenSSL server, potentially containing secret keys.

The vulnerable line of code had been introduced within a larger commit[3] by Robin Seggelmann on 1 January 2012. The whole commit changed 20 files and added 561 lines. Which, not the least, led to a large set of conspiracy theories about the “true” motivation and whether the line has been intentionally been buried. But aside from this, hundreds of people must have reviewed this piece of code and none had spotted the problem although `memcpy` is a known dangerous function, sucessible to memory over-read.

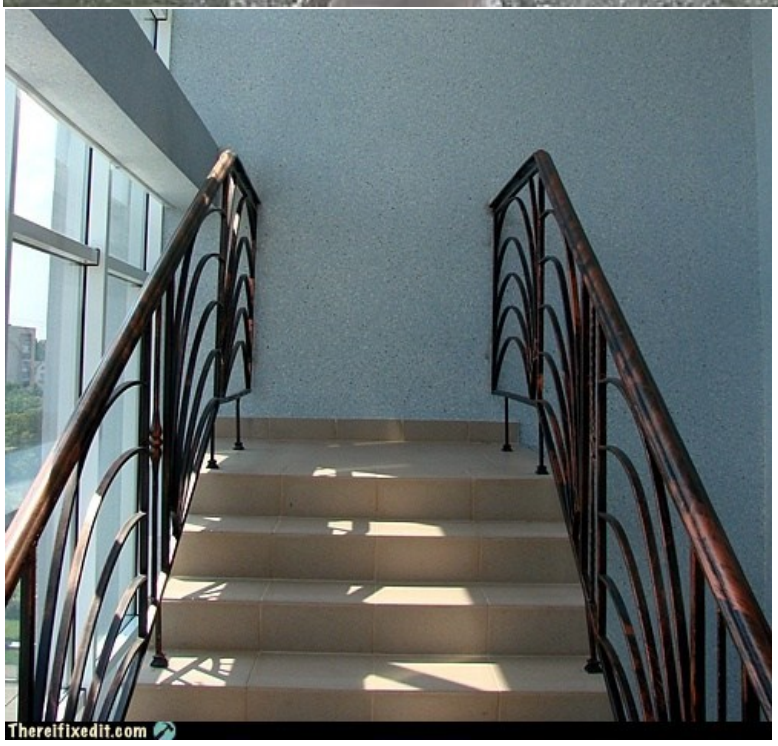
A good summary can be found at `https://www.csoonline.com/article/3223203/what-is-the-heartbleed-bug-how-does-it-work-and-how-was-it-f` `html` (last visited 2020-03-02).

1.6 Security Controls

Wie soll man Sicherheit herstellen? Werden die Maßnahmen zu offen gestaltet, dann können sie umgangen werden und die Schutzziele werden nicht erfüllt. Sind sie zu strikt, dann verhindern sie legitime Nutzung.

[3] `https://github.com/openssl/openssl/commit/bd6941cfaa31ee8a3f8661cb98227a5cbcc0f9f`

Zu offen: 

Zu strikt: 

[Dieser Abschnitt ist aus [5, pg. 218] entnommen und wird, um Übersetzungsfehler zu vermeiden, in deutscher Sprache präsentiert.]

Die Sicherheitsgrundfunktionen bieten einen Baukasten zur Sicherstellung von grundlegenden Sicherheitseigenschaften und finden im Grundschutz insbesondere dann Anwendung, wenn eine niedrige Schadenshöhe zu erwarten ist. Eine vollständige Bedrohungs- und Risikoanalyse stellt in solchen Fällen eine unnötig hohen Aufwand dar. Grundsätzlich sollte jede Sicherheitsstrategie zumindest Konzepte zur Realisierung der Grundfunktionen enthalten.

Die Sicherheitsgrundfunktionen sind:

- Identifikation und Authentifikation
- Rechteverwaltung
- Rechteprüfung
- Beweissicherung
- Wiederaufbereitung
- Gewährleistung der Funktionalität

[5, pg. 218]

1.6.1 Identifikation und Authentifikation

- Abwehr von Maskierungsangriffen (Spoofing)
- Wann und Wer wird authentifiziert?
  - Welche Aktionen erfordern (welche Form der) Identität
  - z.B.: Systemzugang, DB-Zugriff,. . .
- Fehlerbehandlung

1.6.2 Rechteverwaltung

- Welche Subjekte auf
- Welche Objekte unter
- Welchen Bedingungen, mit
- Welchen Rechten?
- Rechtegranularität

1.6.3 Rechteprüfung

- Policy Enforcement Points
- Bsp.: `open` file vs. `read` file
- Ausnahmebehandlung bei unauthorisierten Zugriffsversuchen

1.6.4 Beweissicherung

- Unabstreitbarkeit/Non-Repudiation herstellen
- Welche Ereignisse Wie protokollieren
- Zugriff auf Protokolle?
- Fälschbarkeit der Protokolle?

1.6.5 Wiederaufbereitung

- z.B. Hauptspeicher, Festplatten, USB-Sticks
- Informationen sicher entfernen

1.6.6 Gewährleistung der Funktion

Unterthema Backup:

> "Nobody is interested in Backups, what everybody wants is Restore." Michi Nagorsnik
>
> "Only wimps use tape backup: real men just upload their important stuff on ftp, and let the rest of the world mirror it ;)" Torvalds, Linus (1996-07-20)
>
> Kein Backup — Kein Mitleid!

1.7 Security Law

- NIS Directive [**NIS-Directive-2016**] Europäische Gesetznorm zur Einführung von nationaler Cyber-Sicherheits-Gesetzgebung
- Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme (IT-Sicherheitsgesetz – ITSiG, 2015, Artikelgesetz), beeinflusst (uA)
  - Telekommunikationsgesetz (TKG)
  - Telemediengesetz (TMG)

  - Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSI-Gesetz – BSIG) [**BSIG2015**]

  - Gesetz über die Elektrizitäts- und Gasversorgung (Energiewirtschaftsgesetz – EnWG)

  - ...

- IT-Sicherheitskatalog gemäß § 11 Absatz 1a Energiewirtschaftsgesetz

- Verordnung zur Bestimmung Kritischer Infrastrukturen nach dem BSI-Gesetz (BSI-Kritisverordnung – BSI-KritisV) [**BSI-KritisV2016**, **BSI-KritisV2017**]

- Artikelgesetz

- Telemedien/Telekommunikation nach “Stand der Technik” sichern

- Vorsatz oder fahrlässig:

  - Bußgeld bis 50k€ (KRITIS: 100k€)

  - ggF. persönliche Haftung

- BSiG § 8c: Sicherheitsanforderungen digitale Dienste

  - Maßnahmen: Sicherheit, Kontinuität, Erkennung,...

  - Meldepflicht von Vorfällen

Gesetz
zur Erhöhung der Sicherheit informationstechnischer Systeme
(IT-Sicherheitsgesetz)*

Vom 17. Juli 2015

Der Bundestag hat das folgende Gesetz beschlossen:

Artikel 1

Änderung des
BSI-Gesetzes

Das BSI-Gesetz vom 14. August 2009 (BGBl. I S. 2821), das zuletzt durch Artikel 3 Absatz 7 des Gesetzes vom 7. August 2013 (BGBl. I S. 3154) geändert worden ist, wird wie folgt geändert:

1. § 1 wird wie folgt gefasst:

„§ 1

Bundesamt für
Sicherheit in der Informationstechnik

Der Bund unterhält ein Bundesamt für Sicherheit in der Informationstechnik (Bundesamt) als Bundesoberbehörde. Das Bundesamt ist zuständig für die Informationssicherheit auf nationaler Ebene. Es untersteht dem Bundesministerium des Innern."

2. Dem § 2 wird folgender Absatz 10 angefügt:

„(10) Kritische Infrastrukturen im Sinne dieses Gesetzes sind Einrichtungen, Anlagen oder Teile davon, die

1. den Sektoren Energie, Informationstechnik und Telekommunikation, Transport und Verkehr, Gesundheit, Wasser, Ernährung sowie Finanz- und Versicherungswesen angehören und
2. von hoher Bedeutung für das Funktionieren des Gemeinwesens sind, weil durch ihren Ausfall oder ihre Beeinträchtigung erhebliche Versorgungsengpässe oder Gefährdungen für die öffentliche Sicherheit eintreten würden.

Die Kritischen Infrastrukturen im Sinne dieses Gesetzes werden durch die Rechtsverordnung nach § 10 Absatz 1 näher bestimmt."

3. § 3 wird wie folgt geändert:
   a) Absatz 1 Satz 2 wird wie folgt geändert:
      aa) In Nummer 2 werden die Wörter „zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" durch die Wörter „erforderlich ist, sowie für Dritte, soweit dies zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" ersetzt.
      bb) In Nummer 15 werden die Wörter „kritischen Informationsinfrastrukturen" durch die Wörter „Sicherheit in der Informationstechnik Kritischer Infrastrukturen" und der Punkt am Ende durch ein Semikolon ersetzt.
      cc) Die folgenden Nummern 16 und 17 werden angefügt:
         „16. Aufgaben als zentrale Stelle im Bereich der Sicherheit in der Informationstechnik im Hinblick auf die Zusammenarbeit mit den zuständigen Stellen im Ausland, unbeschadet besonderer Zuständigkeiten anderer Stellen;
         17. Aufgaben nach den §§ 8a und 8b als zentrale Stelle für die Sicherheit in der Informationstechnik Kritischer Infrastrukturen."
   b) Folgender Absatz 3 wird angefügt:

„(3) Das Bundesamt kann Betreiber Kritischer Infrastrukturen auf deren Ersuchen bei der Sicherung ihrer Informationstechnik beraten und unterstützen oder auf qualifizierte Sicherheitsdienstleister verweisen."

4. Die Überschrift von § 4 wird wie folgt gefasst:

* Notifiziert gemäß der Richtlinie 98/34/EG des Europäischen Parlaments und des Rates vom 22. Juni 1998 über ein Informationsverfahren auf dem Gebiet der Normen und technischen Vorschriften und der Vorschriften für die Dienste der Informationsgesellschaft (ABl. L 204 vom 21.07.1998, S. 37), zuletzt geändert durch Artikel 26 Absatz 2 der Verordnung (EU) Nr. 1025/2012 des Europäischen Parlaments und des Rates vom 25. Oktober 2012 (ABl. L 316 vom 14.11.2012, S. 12).

Das Bundesgesetzblatt im Internet: www.bundesgesetzblatt.de | Ein Service des Bundesanzeiger Verlag www.bundesanzeiger-verlag.de

- BSI Gesetz
  - Einführung ISBÖFI (Infrastruktur im besonderem öffentlichen Interesse)
    - e. g., Rüstungsindustrie
  - Anzeigepflicht beim Einsatz kritischer Komponenten.Nach §2 Abs. 13 solche Komponenten die wesentlich für die Erbringung der Dienste in kritischen Infrastrukturen sind. In einer strengen Auslegung können das sehr viele Komponenten sein. Das BMI hat Erlaubnisvorbehalt.
  - Geheimhaltung von Schwachstellen, wobei das BSI hier auf Weisung des BMI handeln müsste. Zerstört nach Ansicht vieler Experten das Vertrauensverhältnis von White-Hat Sicherheitsforschern und BSI.

- ...

Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSiG)

- §§ 4 und 8b: Zentrale Meldestelle (Bund, KRITIS)
- § 5: Gefahrenabwehr für Bund TK
- § 5a: Incidence Response (Bund, KRITIS) in herausgehobenen Fällen
- § 7a: Produktuntersuchung
- § 8c: Sicherheits und Meldepflicht
- § 9: Zertifizierung, Personen und Systeme
- § 14: Bußgelder 50k/100k€

Aside from appealing to your reason, there are stronger arguments for good behaviour. They are called laws and can be thought of as formal ethics with teeth. If you do not behave according to them you will be punished (if caught obviously, but watch the news who gets caught?). Thus, read the following paragraphs carefully and realise where something could go awry.

§ 202 StGB: Ausspähen

> § 202a Abs. 1: [Unbefugter Zugang zu Daten: bis 3 Jahre]
>
> § 202b: [Unbefugtes Abhören: bis 2 Jahre]
>
> (1) Wer eine **Straftat** nach § 202a oder § 202b **vorbereitet**, indem er [...] 2. **Computerprogramme**, deren **Zweck** die Begehung einer solchen Tat ist, **herstellt**, sich oder einem anderen **ver-**

> **schafft**, **verkauft**, einem anderen **überlässt**, **verbreitet** oder sonst **zugänglich macht**, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft.

§ 303a StGB: Datenveränderung

(1) Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.

(2) Der Versuch ist strafbar.

(3) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

§ 303b StGB: Computersabotage

(1) Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, dadurch erheblich stört, dass er

1. eine Tat nach § 303a Abs. 1 begeht,
2. Daten (§ 202a Abs. 2) in der Absicht, einem anderen Nachteil zuzufügen, eingibt oder übermittelt oder
3. eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert,

wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Handelt es sich um eine Datenverarbeitung, die für einen fremden Betrieb, ein fremdes Unternehmen oder eine Behörde von wesentlicher Bedeutung ist, ist die Strafe Freiheitsstrafe bis zu fünf Jahren oder Geldstrafe.

(3) Der Versuch ist strafbar.

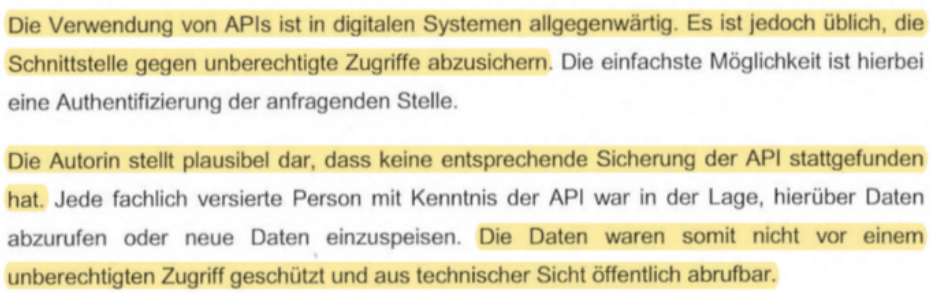
(4) In besonders schweren Fällen des Absatzes 2 ist die Strafe Freiheitsstrafe von sechs Monaten bis zu zehn Jahren. Ein besonders schwerer Fall liegt in der Regel vor, wenn der Täter

1. Vermögensverlust großen Ausmaßes herbeiführt,
2. gewerbsmäßig oder als Mitglied einer Bande handelt, die sich zur fortgesetzten Begehung von Computersabotage verbunden hat,
3. durch die Tat die Versorgung der Bevölkerung mit lebenswichtigen Gütern oder Dienstleistungen oder die Sicherheit der Bundesrepublik Deutschland beeinträchtigt.

(5) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

2021 wurde Lilith Wittmann von der damals an der Regierung befindlichen Partei polizeilich angezeigt, weil sie eine Sicherheitslücke in deren Wahlkampf-App CDU-Connect gemeldet hatte. In ihrem Blog dokumentiert die Sicherheits-

forscherin den Prozess bis zur Einstellung des Verfahrens. Die Ankläger waren auf die Sicherheitslücke aufmerksam geworden, weil die Sicherheitsforscherin selbst sie, entsprechend der Vorgehensweise *Responsible Disclosure*, vor der Veröffentlichung der Schwachstelle informiert hat.



Die Verwendung von APIs ist in digitalen Systemen allgegenwärtig. Es ist jedoch üblich, die Schnittstelle gegen unberechtigte Zugriffe abzusichern. Die einfachste Möglichkeit ist hierbei eine Authentifizierung der anfragenden Stelle.

Die Autorin stellt plausibel dar, dass keine entsprechende Sicherung der API stattgefunden hat. Jede fachlich versierte Person mit Kenntnis der API war in der Lage, hierüber Daten abzurufen oder neue Daten einzuspeisen. Die Daten waren somit nicht vor einem unberechtigten Zugriff geschützt und aus technischer Sicht öffentlich abrufbar.

Figure 1.8: Aktenvermerk zur von Lilith Wittmann gemeldeten Sicherheitslücke in CDU-Connect. Aus der Verfahrensakte, Quelle: Wittmann

Die Ermittlungen der Polizei und Staatsanwaltschaft haben dann ergeben, dass es sich alleine deshalb nicht um eine Straftat nach §202 StGB handeln kann, weil die Daten technisch “nicht vor unberechtigtem Zugriff geschützt”, sondern “öffentlich abrufbar” waren.

Im Nachgang wird eine Bewertung des Datenschutzes erwartet um Einzuschätzen inwieweit die öffentliche Bereitstellung persönlicher Daten durch die Anwendung CDU-Connect gegen die Datenschutzgrundverordnung verstoßen hat.

Exercises

Ex. 1 — Syllabus

1. Welche Zugangsvorraussetzungen authorisieren Sie einen erfolgreichen Schein im Kurs “IT-Sicherheit” zu erlangen? Schreiben Sie ein kurzes Skript, welches nach Eingabe der einzelnen Punktzahlen eine Abschlussnote berechnet.

Ex. 2 — Es raucht, wo ist das Feuer

1. Recherchieren Sie mit Ihrer Gruppe für das nächste Plenum eine Liste von sieben Sicherheitsvorfällen (i. e., Angriffe auf IT oder mit den Mitteln von IT). Bewerten Sie jeweils die Quellenlage, die Schadenshöhe, die Angriffskomplexität und benennen Sie Angriffswege und Schwachstellenursache, betroffene Anlagen/Daten sowie das Datum des Vorfalls. Im Plenum werden jeweils Punkte für die Gruppen verteilt, welche die frühesten, aktuellsten, aufwändigsten und schädlichsten Vorfälle finden konnten. Das Plenum bewertet die Vorfälle und kann Vorschläge aufgrund mangelhafter Quellen und Belege von der Bewertung ausschließen.
   Führen Sie die Vorfälle in einer Tabelle mit den Spalten: Datum, Angreifer, Ziel, Beschreibung, Angriffsform, Klasse des Ziels, Angriffsklasse, Land/Ort, Schadenshöhe, Komplexität, und Links.
   Incident Steckbrief:
   Name Vorfallname
   Datum Jahr des Vorfalls
   Betroffen Wer oder Was war betroffen?
   Impact
   Angriffsvektor
   Attribution
   Beispiel:
   Name Triton/TRISIS
   Datum 2017
   Betroffen Petrochemische Industrieanlage, Saudi Arabien
   Impact Ausgeschaltete Schutzvorrichtungen
   Angriffsvektor Infektion of OT über IT Infrastruktur
   Attribution

Ex. 3 — Definition IT-Sicherheit

1. Entwerfen Sie eine kurze Bewertung der Sicherheitsziele einer demokratischen Wahl. Nutzen Sie dafür die erweiterte Terminologie der Sicherheitsziele. Zuerst müssen sie dafür ein kurzes Modell des Wahlprozesses entwerfen. Nutzen Sie dafür die Akteure: Wähler, Wahlleiter und -mitarbeiter, und Kandidaten. Nutzen Sie als Assets mindestens: Wahlurne, Ergebnisdokument.

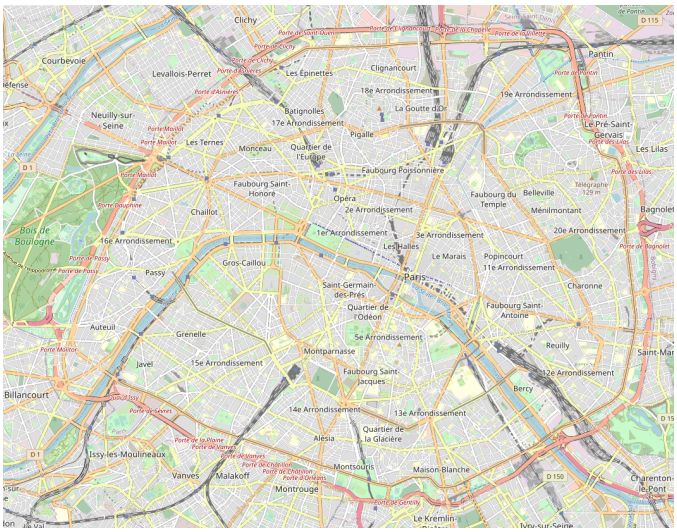
Erarbeiten Sie eine grafische Darstellung des Wahlprozesses aus der die Interaktionen und Abhängigkeiten der Akteure und Assets erkenntlich werden. Formulieren Sie Sicherheitsziele für Assets und Interaktionen (Prozessschritte) in tabellarischer Form, sortiert nach Assets und Interaktionen. Beschreiben Sie die Aufgaben der Akteure im Wahlprozess wo diese die Erfüllung von Sicherheitszielen betreffen. Erweitern Sie die Liste der Akteure um potentielle externe Angreifer und identifizieren Sie in der erweiterten Liste der Akteure und Angreifer Mißbrauchsmöglichkeiten.
Sie sollen die Anforderungen so konkret formulieren, dass für jeden Schritt und jedes Asset klar wird welche Eigenschaften umgesetzt werden müssen.

2

Network Security Architecture

2.1 Zones and Conduits

An old and proven principle of physical and digital security is *separation*. It can be found in the Saltzer and Schröder Security principles as well as in the architecture of castles, office buildings or law.



2.1.1 Network Segmentation

In networking, separation

- Significant issue hostile/unwanted trespass
  - from benign to serious
- User trespass

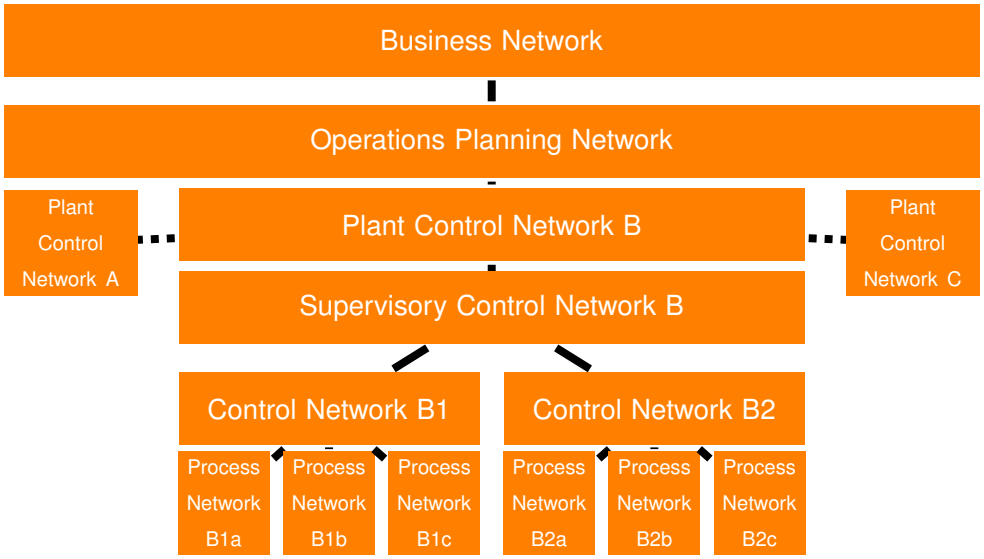


Figure 2.1: Conceptual representation of network segmentation. Based on [**knapp2015ics**].

  - unauthorized logon, privilege abuse
- Software trespass
  - virus, worm, or trojan horse
- Classes of intruders:
  - Masquerader: (Outside) user, who penetrates a system's access control to exploit user accounts
  - Misfeasor: Insider user, who accesses data, programs, or resources unauthorized, or who misuses authorized access
  - Clandestine user: Manipulates supervisory control of system, e. g., to evade auditing and access controls

2.2 Firewalls

- Firewalls
  - prevention of intrusion
  - separation of segments
- Firewall Architecture

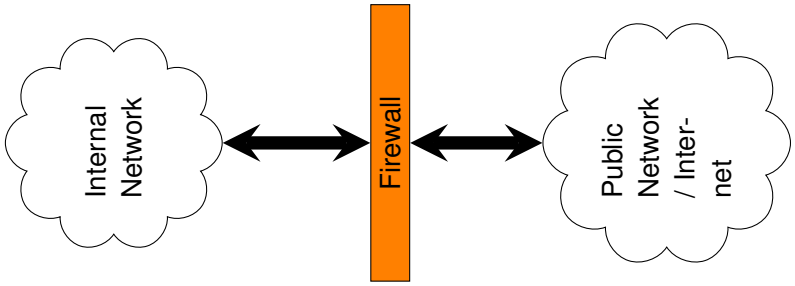


Figure 2.2: Firewall Principle

  - Layered Systems
  - Functional/Criticality
- Intrusion Detection
  - Eyes and Ears
  - Host and Network

A firewall is a set of one or more components, of hard- and software, through which all traffic between an inside network and and outside network is directed.

Firewall "An internetwork gateway that restricts data communication traffic to and from one of the connected networks (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall).

[...] A firewall is not always a single computer. For example, a firewall may consist of a pair of filtering routers and one or more proxy servers running on one or more bastion hosts, all connected to a small, dedicated LAN (see: buffer zone) between the two routers. The external router blocks attacks that use IP to break security (IP address spoofing, source routing, packet fragments), while proxy servers block attacks that would exploit a vulnerability in a higher-layer protocol or service. The internal router blocks traffic from leaving the protected network except through the proxy servers. The difficult part is defining criteria by which packets are denied passage through the firewall, because a firewall not only needs to keep unauthorized traffic (i.e., intruders) out, but usually also needs to let authorized traffic pass both in and out." [**Shirey07InternetSecurityGlossary**]

Also see "guard" in [**Shirey07InternetSecurityGlossary**]

Filtering Router "An internetwork router that selectively prevents the passage of data packets according to a security policy." [**Shirey07InternetSecurityGlossary**]

Packet Filter A firewall working on ISO/OSI-layers 3 and 4 (and sometimes 2 as well).

Proxy Firewall A proxy firewall acts as intermediary at ISO/OSI-layers 3 and 4. The proxy is presenting itself as the server to the exterior world, usually accepts the communication and may provide additional security services, like authentication, access control, auditing.

Usually the requests to the service are then posted by the firewall to the service provider, this means that the service will only receive communication from the firewall. Communication may also be re-packed, eg. into different network protocols.

Application Firewall A application firewall provides an application specific intermediary that shields the interior application by acting instead of this application to the exterior. An application firewall understands the application protocols and may re-encode or otherwise "clean" the requests. This allows the firewall to undertake application specific analysis.

Bastion Host "A strongly protected computer that is in a network protected by a firewall (or is part of a firewall) and is the only host (or one of only a few) in the network that can be directly accessed from networks on the other side of the firewall." [**Shirey07InternetSecurityGlossary**]

2.3 Firewall Architecture

A good explanation is found at: `http://www.invir.com/int-sec-firearc.html`

2.3.1 Subnetworking Fundamentals

We would hope that this is already well known to any computer related person.

An Internet address, with a few exceptions, consists of a *network* and a *host* part. The host part is defined by a host mask where the binary 1es define which bits are included in the host part of an address.

Thus, different sub-networks are differentiated by different network parts of the address. Usually the network part consists of a sequence of bits at the beginning of an address and the host part is the remainder.

IPv4 Address Example

32 Bit: 141.99.96.123

Hostmask: $\underbrace{11111111.11111111.11111111}_{network}.\underbrace{00000000}_{host}$

Hostmask/24 255.255.255.0

Hostmask/16 255.255.0.0

Hostmask/22 255.255.253.0

IPv4 Address Registry[1] defines, in summary:

0.0.0.0 Network Address

10.0.0.0/8 Private Use

78.0.0.0/8 RIPE NCC

127.0.0.0/8 Loopback [RFC 1122]

192.168.0.0/16 Private-Use Networks [RFC 1918]

224.0.0.0/8 Multicast

255.255.255.255 Limited Broadcast

IPv6 Addresses

IPv6 on this level is not very much different, except that the addresses are longer and we separate 4-byte chunks by ':' in common notation. IPv6 uses 128 Bit Addresses. Beyond that, the differences between IPv4 and IPv6 become a little to much for this small reminder.

2.3.2 Compartmentalisation

Please also see [**CZ95BuildingInternetFirewalls**] for more information.

Screening Router

Dual-Homed Host

Screened-Host Firewall

Screened-Subnet Firewall

Compartmentalisation means the separation

De-Militarised Zone (DMZ)

[1] `http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml`

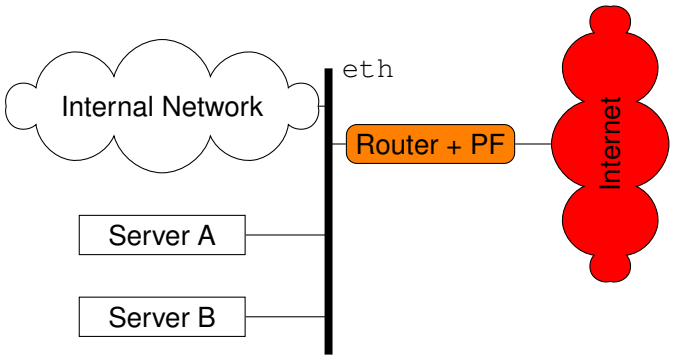


Figure 2.3: Screening Router

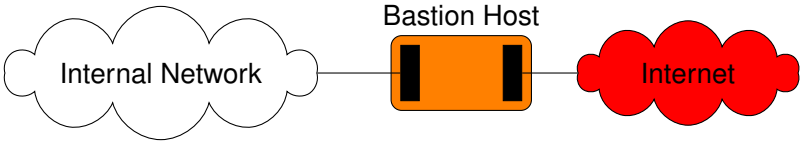


Figure 2.4: Screened-Host Firewall

Screening Router

Dual-Homed Bastion

Screened-Host Firewall

Screened-Host Firewall is a firewall architecture, where the routing firewall screens single hosts. Public servers, application firewall and internal network are situated on the same network segment (ethernet segment). This means, that the internal network is not separated from the internal network by a firewall.

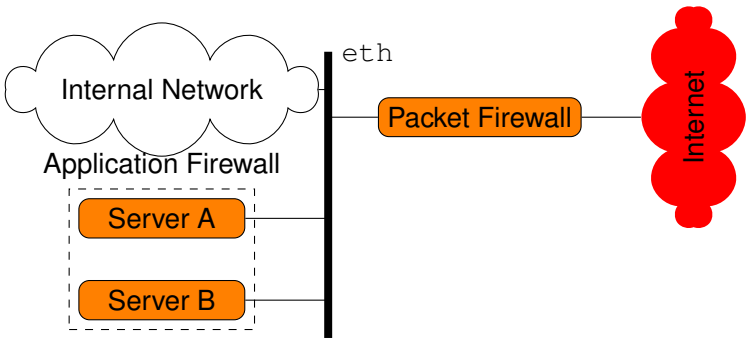


Figure 2.5: Screened-Host Firewall

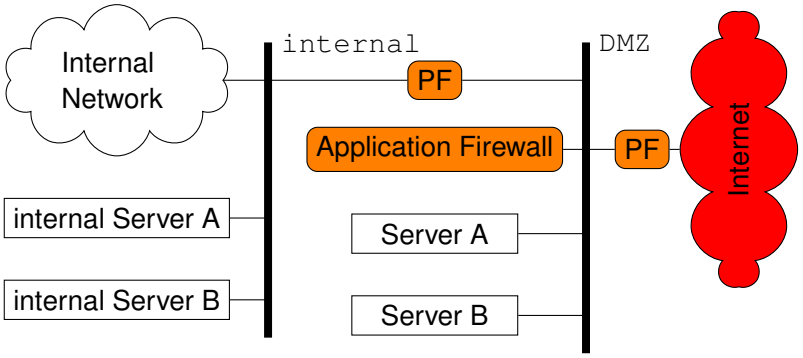


Figure 2.6: Screened-Subnet Firewall

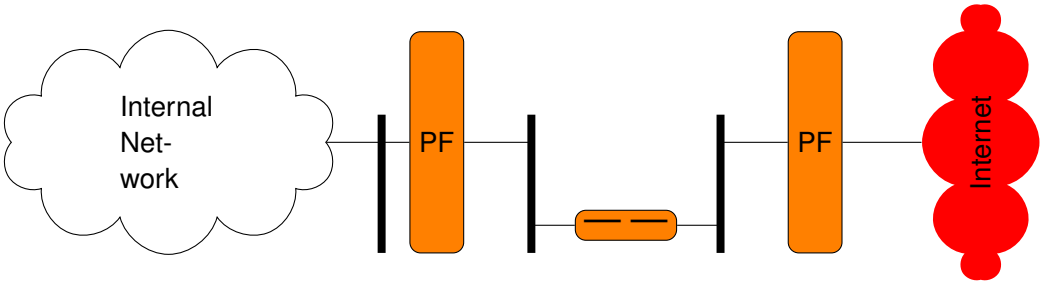


Figure 2.7: Split-Screen Firewall (Belt-and-Suspenders)

Screened-Subnet Firewall

A screened-subnet firewall goes one step further in that it has an additional firewall that separates internal from public networks.

Split-Screen Subnet

A Screened Subnet architecture with an additional Dual-Homed Bastion Host that splits the Demilitarised Zone into two separate sub-networks.

(also called Belt-and-Suspenders)

2.3.3 Packetfilter Firewalls

A Packetfilter Firewall is working on individual network packets (i. e., IP-packets including layer 4 headers).

Stateless A stateless filter can make descisions based only on the contents of the currently inspected packet. It can not include information gathered from previous packets into its decision — that is, because it has no state (i. e., memory).

Stateful A stateful firewall has a memory and can thus dynamically adapt its current decisions based on previous events. It could, for example, allow an incoming packet of a TCP-connection, if the establishment of that connection has previously been observed.

```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp --dport 80,443 -j ACCEPT
iptables -A FORWARD -p tcp --sport 80,443 -j ACCEPT
```

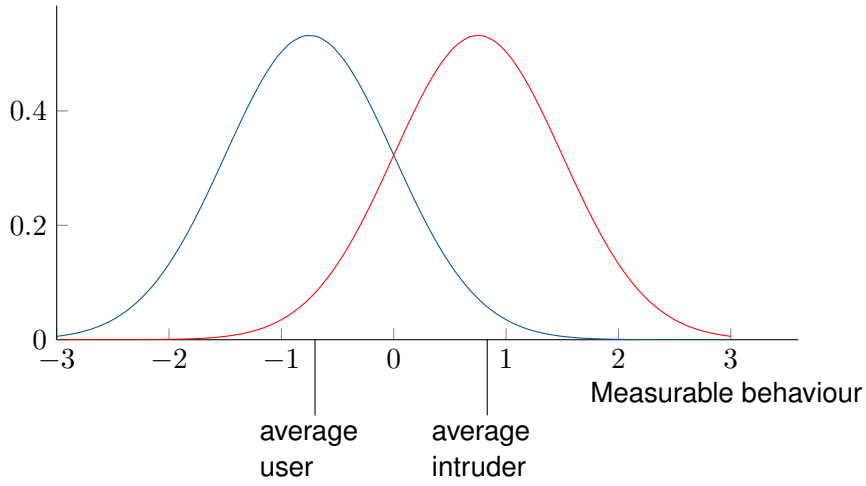
```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp --dport 80,443 \
             -s 192.168.1.0/24 -j ACCEPT
iptables -A FORWARD -p tcp -m state \
     --state ESTABLISHED,RELATED -j ACCEPT
```

2.4 Intrusion Detection

- Network: e. g., Snort
- Host: e. g., Tripwire
- Observations of non-attacked systems
  1. User, process actions conform to *statistically predictable* pattern
  2. User, process actions do not include sequences of actions that subvert the security policy
  3. Process actions correspond to a set of *specifications* describing what the processes are allowed to do
- As sumption underlying intrusion detection
  - Systems under attack do *not* meet at least one of the points
- Intrusion detection models:
  - Anomaly detection
  - Signature detection
- Intrusion detection structures:
  - Host-based IDS: monitor single host activity
  - Network-based IDS: monitor network traffic
- Logical components:
  - Sensors - collect data

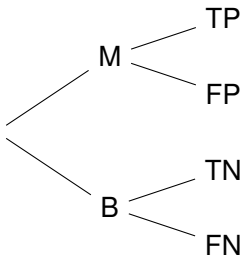
  - Analyzers - determine if intrusion has occurred
  - User interface - manage / direct / view IDS
- Input of IDS: Audit records

2.4.1 Anomaly-based IDS



Extremely rare events — Many False Errors

Accuracy 99% (i. e., for every 100 benign events there will be 1 *false positiv*.



Assume: 0.95 of traffic is benign (i. e., 5% is malicious)

$$
\begin{aligned}
P(TP) &= 0.95 \cdot 0.99 = 0.94 \\
P(FP) &= 0.95 \cdot 0.01 = 0.009 \\
P(TN) &= 0.05 \cdot 0.99 = 0.04 \\
P(FN) &= 0.05 \cdot 0.01 = 0.0005
\end{aligned}
$$

400.000 events/day $\Rightarrow$ 3.600 False Alarms (200 undetected intrusions)

Statistical Moments This term (used in the slides) means the mathematical term of moments. First and Second Moment are probably better known as *mean* and *variance* by the audience.

2.4.2 Signature-based IDS

```
# alert tcp $HOME_NET 2589 -> $EXTERNAL_NET any
  ( msg:"MALWARE-BACKDOOR - Dagger_1.4.0";
    flow:to_client,established;
    content:"2|00 00 00 06 00 00 00|Drives|24 00|",depth 16;
    metadata:ruleset community;
    classtype:misc-activity; sid:105; rev:14; )
```

- Intern to Extern
- TCP port 2589 to any port
- Contains "`2|00 00 00 06 00 00 00|Drives|24 00|`"

```
# alert tcp $EXTERNAL_NET any -> $HOME_NET 7597 ( msg:"MALWARE-BACKDO
# alert tcp $EXTERNAL_NET any -> $HOME_NET 12345:12346 ( msg:"MALWARE-
```

Exercises

Ex. 4 — Network Exploration in der Übungsstunde. Durchführung auf dem Sec-Docker-Image

.

1. Wer befindet sich in Ihrem lokalen Ethernet Segment? Nutzen sie `netdiscover` und beobachten und loggen Sie die Exploration ihrer Kollegen.

Ex. 5 — Segmenting your Network

1. Legen Sie mit `brctl` Netzwerksegmente (Ethernet Bridges) für die Ebenen *SCADA*, *Control*, und *Process* einer gedachten industriellen Anlage an und verbinden Sie diese über die `iptables` Konfiguration jeweils derart, dass
   - jedes Netz einen eigenen IP-Bereich aus dem 10'er Netz hat
   - alle ausgehenden Verbindungen aus dem Prozessnetz maskiert werden,
   -

3

Authorisation and Access Control

4

Access Control

This chapter is presenting selected mechanisms for authorisation and management of access rights.

The first computers where resources that where protected for themselves by keeping them in secured areas. Anyone who could access these areas (and had the knowledge to operate them) could use these computers. Access control was implemented in classical ways that restricted physical access. The most fundamental method probably is a door-lock. In this chapter we will concern ourselves with the access control methods that came with multi-user systems, networked computers and digital objects.

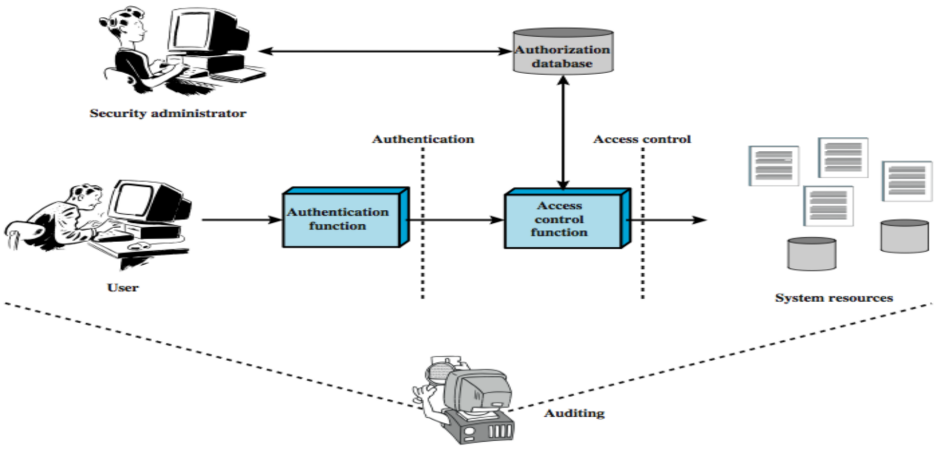
Access control allows or denies access to given resources by a given principal in a defined way. It requires authentication to establish the identity of the principal. It consists of an access control system, that makes decisions based on an access control database. In professional context, it further is completed by a separate auditing system.

4.1 Access Control Components

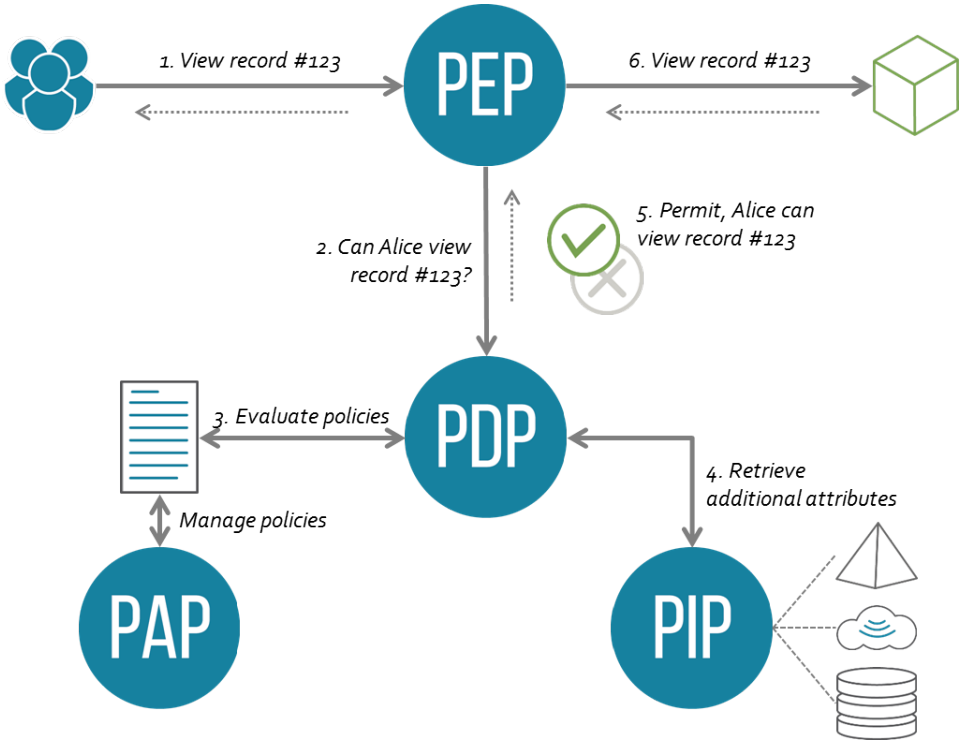
Authentication function: Verification of claimed identity (See Chapter "Authentication")

Access control function: Determines if specific requested access of user is permitted based on *authorisation* database.

Audit: Monitors and keeps record of user accesses



Let us take a closer look at the components of an access control function:



PAP (Policy Administration Point) Point which manages access authorization policies

PDP (Policy Decision Point) Point which evaluates access requests against authorization policies before issuing access decisions

PEP (Policy Enforcement Point) Point which intercepts user's access request

to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision

PIP (Policy Information Point) The system entity that acts as a source of attribute values (i. e., a resource, subject, environment)

PRP (Policy Retrieval Point) Point where the XACML access authorization policies are stored, typically a database or the filesystem.

Discretionary Access Control (DAC) ("discretionary": (de) „Handlungsfreiheit", „Ermessens. . . ")

Mandatory Access Control (MAC) ("mandatory": (de) „obligatiorisch", „zwingend")

Role-based Access Control (RBAC): Access Control based on roles instead of subjects. Is the most common type of access control in medium to large organisations.

4.2 Privilege-Separated Operating System

This section is a stub.

Privilege-Separated Operating System

- distinct identities (e. g., user,group,. . .)
- processes/applications related to identity
- e. g., Android (Linux), current Windows

We are talking about android[1].

4.3 Access Control Function

The Access Control Function embodies the decision of whether an action is allowed or not in a given execution context.

4.4 Process Space Protection

This topic is intentionally left out. Please refer to a lecture on operating systems.

[1] `https://developer.android.com/guide/topics/security/permissions.html`, 2013-07-02

4.5 Discretionary Access Control

4.6 Access Control Matrix

An *Access Control Matrix* (ACM) is a representation of access rights in discretionary access control. It consists of a list of all subjects and all objects represented in rows and columns and thus provides a complete model of all access rights. The fields of the matrix hold the access rights of the subject (in our examples found in the rows) to an object (in our examples found in the columns).

- Policy representation for discretionary based authorization
- Let
  - S be set of all subjects,
  - O be set of all objects and
  - R be set of all operation rights.
- Access Control Matrix (ACM) M defines for pairs $(s \in S, o \in O)$ set of possible operations $M_{s,o} \subseteq R$.

	o_1	o_2	o_3
s_1	rw	w	r
s_2		rw	rwx

4.6.1 Access Rights

It is not difficult to creatively define a whole heap of different rights. The question is, how much granularity you need. In the following we produced a summary list of the most common rights.

Owner is a “special” right that describes that a subject has all rights on an object (because it owns it).

Read/Write/Execute are the classic rights you probably know from using any standard Unix-like system. Reading and writing describe the rights to read data from an object and modify or add to its contents. By execution we describe the right to run the object as a program.

Append is a right rarely found, that describes, that one is not allowed to alter existing contents of an object, but may only add contents at the end of the object.

Create/Delete are slightly different from *Write* in that they allow to completely create or remove an object and not only its contents. This includes cre-

ation and removal in the ACM. Often *Create* and *Delete* are implemented as the right to *Write* of a directory-like object.

Search means the right to display contents of a directory. (This is again often represented by *Read* on the directory.)

The most common implementations have Read, Write and Execute rights and implement Create, Delete and Search as Read and Write on special directory-objects.

4.6.2 Operations on ACM

It is sensible to formally define operations on an Access Control Matrix to allow for formal analysis of state transitions. The operations either delete or create subjects (rows), objects (columns), or rights (in fields).

An operation is defined by pre-conditions that must be given before an operation may take place, and post-conditions that specify the effect of the operation. For the set of subjects S, of objects O, of rights R, and an ACL A. A system state is described by the triple (S, O, A). An operation c defines a state transition $(S, O, A) \underset{c}{\mapsto} (S', O', A')$:

add subject s **Pre-Condition:** $s \notin S$

Post-Condition:
- $S' := S \cup \{s\}$ and $O' := O \cup \{s\}$ (create user)
- $\forall y \in O' : A'[s, y] := \varnothing$ and $\forall x \in S' : A'[x, s] := \varnothing$ (empty rights by default)
- $\forall x \in S, \forall y \in O : A'[x, y] := A[x, y]$ (copy existing)

delete subject s **Pre-Condition:** $s \in S$

Post-Condition:
- $S' := S \backslash \{s\}$ and $O' := O \backslash \{s\}$ (delete user)
- $\forall x \in S', \forall y \in O' : A'[x, y] := A[x, y]$ (copy remaining)

create object o **Pre-Condition:** $o \notin O$

Post-Condition:
- $O' := O \cup \{o\}$ and $S' := S$ (create object)
- $\forall y \in S' : A'[y, o] := \varnothing$ (empty rights by default)
- $\forall x \in S, \forall y \in O : A'[x, y] := A[x, y]$ (copy existing)

delete object o **Pre-Condition:** $o \in O$ and $o \notin S$

Post-Condition:
- $O' := O \backslash \{o\}$ and $S' := S$ (delete object)
- $\forall x \in S', \forall y \in O' : A'[x, y] := A[x, y]$ (copy remaining)

add right r **to** (s, o) **Pre-Condition:** $s \in S$, $o \in O$, and $r \notin A[s, o]$

Post-Condition:
- $S' := S, O' := O$ (copy subjects)
- $A'[s, o] := A[s, o] \cup \{r\}$ (redefine matrix cell)
- $\forall x \in (S \backslash \{s\}) : \forall y \in (O \backslash \{o\}) : A'[x, y] := A[x, y]$ (copy existing)

delete right r **from** (s, o) **Pre-Condition:** $s \in S$, $o \in O$, and $r \in A[s, o]$

Post-Condition:
- $S' := S, O' := O$ (copy subjects)
- $A'[s, o] := A[s, o] \backslash \{r\}$ (redefine matrix cell)
- $\forall x \in (S \backslash \{s\}), \forall y \in (O \backslash \{o\}) : A'[x, y] := A[x, y]$ (copy existing)

There are additional operations that could be defined on ACM, for example, *transfer* or *grant* of rights.

4.6.3 ACM Implementation

Issues of access control matrix:

Memory Intensive ACM are not suitable for implementation on large systems with many subjects or objects, i. e., they require $|S \times O|$ fields.

Complex Lookup ACM are not stored with either subjects or objects, i. e., they have to be separate from the objects concerned by them, which requires proceedures and storage additional to object-access for the lookup.

Furthermore, a large structure where individual entries are rarely accessed is not efficient to be kept in memory, thus lookup-strategies have to be implemented, which increase the complexity of the system.

Sparse Matrix Many entries in matrix will be empty, but memory is nevertheless allocated or complex algorithm (e. g., hash-tables) have to be deployed.

$\Longrightarrow$ Access control matrix is only abstract concept

Solution:

- Ignore empty matrix entries, i. e., no memory allocation for subject s without rights on object o.
- Store rights with subject or object:

  Capability List (CL) is a row-wise implementation of an ACL. Defines the rights of a subject to a list of objects. Can be compared to a credential that can be used to claim rights to objects, e. g., the paperwork that proves your ownership of a car.

Access Control List (ACL) a list of access rights, related to given objects. Column-wise implementation of an ACL. Can be compared to a VIP-list at a concert, where the concert relates to an object and the list enumerates the subjects that have access right.

Capability Lists

You may compare capabilities to tickets that allow you entry to a concert. A capability is kept with the subject that enjoys the access rights granted by that capability.

- Each subject has a capability, which is an *unforgeable* token
- Capability corresponds to list of access control entries (row of Martrix)
- Each entry contains an object o and the rights of s on object o
- Subject can create object and grant other subjects capabilities on object.
- Subject s may be allowed to pass parts of capability to other subject s' , e. g., in case that s invokes a process s'.

Issues:

- Who may read this file? It is difficult to find out which subject has access on which object.
- Revocation is difficult: Hard to revoke capability, because difficult to track transfer of access to other subjects.

ACM

	o_1	o_2	o_3
s_1	rw	w	r
s_2		rw	rwx

Capabilities $s_1 \longrightarrow [(o_1, rw), (o_2, w), (o_3, r)]$

Access Control List (ACL)

- Each object o is associated with list of access control entries (column of matrix).
- Each entry contains a subject and its rights on object o

Unix Access Control

Operation systems often implement a hybrid approach to access control utilising both Access Control List (ACL) and Capability Concepts.

In Unix-Systems, the primary access control model, represented in the file-system, is implemented as an ACL. But the call `open` provides processes with a file-descriptor, which is used to access the file without further verification of the ACL in the file system. File-descriptors thus resemble temporary capabilities, i. e., they are attached to the process acting on behalf of a subject.

The Unix-model provides a small set of rights (read, write, execute) but also an extension to the subject model. Each object has rights assigned for the "owner" of the file, which is an identity from the systems subject list. Additionally the Unix ACL also defines rights for a "group" of subjects and a different set for the rest of the "world". The group is a set of user identities managed in a special system group file.

DAC Safety

The main problem of ACM-based systems is to decide if there is a sequence of transitions that may lead to an undesired state, given the operations on an ACM.

The first problem is to define what states are desired and which are undesired.

Safety question: Is it possible to reach a state within which a right can be given or an operation executed that is not "intended"? For example: Can an attacker grant access rights to an object without consent of the owner of that file? A system is considered safe, if it can be proven for every safe state that there exists no sequence of commands[2] that lead to unsafe states.

mono-operational Commands that are only mapped onto single primitive operations. For example there is no command that both creates a file and changes a right. Which implies that all newly created objects have the same rights.

primitive operation operations that only produce single effects

4.7 Mandatory Access Control

Mandatory Access Control provides a concept for systemic control of resources. Different from DAC, some baseline access rules are set by the access control system an cannot be changed by users (or administrators). Often the objective is to provide control over information flow.

[2] i. e., state changes

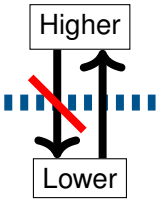


Figure 4.1: Bell-LaPadula Information Flow Directive

4.7.1 Bell-La Padula

The Bell-LaPadula model aims at flow control, by preventing information flow from higher security levels down to lower security levels.

Security Levels $L : S \cup O \rightarrow$

Top Secret

Secret

Confidential

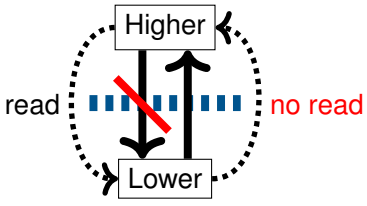
Unclassified

Objects have *security classification* $L(o)$

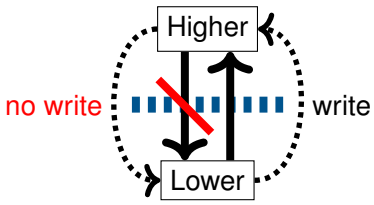
Subjects have *clearance levels* $L(s)$

No information shall flow downwards!

1. Simple Security Property/no-read-up rule— no subject of a lower security clearance may read from an object of a higher security classification
2. *-Property/no-write-down rule— no subject of a higher security clearance may write to an object with lower security classification.
3. Discretionary Security Property— an additional ACM implementation is used to further restrict access.



Subject s can read from object o if $L(s) \geqslant L(o)$ AND Discretionary Access is granted



Subject s can write to object o if $L(s) \leqslant L(o)$ AND Discretionary Access is granted

Summary and Discussion

- Blind writing: A user may not read the effects of the modifications he just has made to a file, if he has lower clearance. This is problematic with respect to the next point.
- Low integrity protection: Subjects are able to blindly write upwards. As subjects with lower clearance are able to write to objects with higher classification, the multi-layer security model do not increase protection against manipulation.
- Covert Channels (see following slide)

4.7.2 Covert Channels

It is actually very hard, if not impossible to completely control all information flows.

2-Phase Locking Exploit

2-Phase locking is a method for concurrency control[3]. Use file-locking for communication:

- Given s_h, s_l, o_l: Subjects with high and low clearance and object with low classification.
- s_h may read o_l, s_l may write
- Read operation triggers lock
- Essential binary signal from high to low.

[3] https://en.wikipedia.org/wiki/Two_phase_locking

4.8 Role-based Access Control

The intention of Role-based access control is to simplify the task of managing access rights by providing functional roles. In professional context, the obvious question before granting a right is: For what is it necessary that an individual is able to do this? And the obvious generalised answer almost always will be: to fulfil its function. Thus the better way to manage access rights is to model the access rights of functions and then simply assign functions to individuals. This way of dealing with rights seems obvious and is probably used for ages outside the digital world.

Imagine system with 1000 subjects, 100000 objects and 10 access rights, there are $1000 \times 100000 \times 10 = 10^9$ authorization triples

- Reducing complexity to maintain access control lists like in CL and ACL by introducing intermediate layer between subject and object called roles
- Access rights are assigned to roles instead of to single subject, thus subject accessing an object must have corresponding role
- Subject can have several roles
- There are less roles than users, thus reducing access control overhead

User an individual (with access to the system)

Role a named job function. A role definition should describe the responsibility and authority of any user assuming this role.

Permissions the access rights of a role.

Session is a temporally valid mapping from user to a set of activated roles.

Constraints restrict the accepted types of sessions, i. e., which roles are allowed to be active at the same time (dynamic constraint). Can also define roles that may not be assigable to the same user (static constraint), i. e., the function behind the roles should not be combined in the same user.

[**Stallings11CryptographyandNetwork**]

Figure 4.2 shows the mapping of users (subjects) to roles, which are then mapped onto access rights on objects. Roles take up the position of subjects in the ACM model, which is beneficial if the number of roles is smaller than the number of users.

Furthermore, Roles allow to model access rights based on function of, i. e., the actual requirements of a position (job) with associated tasks. In practise this allows for more generic access right assignments, makes the idea behind the assignments much more transparent and provides long-term solutions.

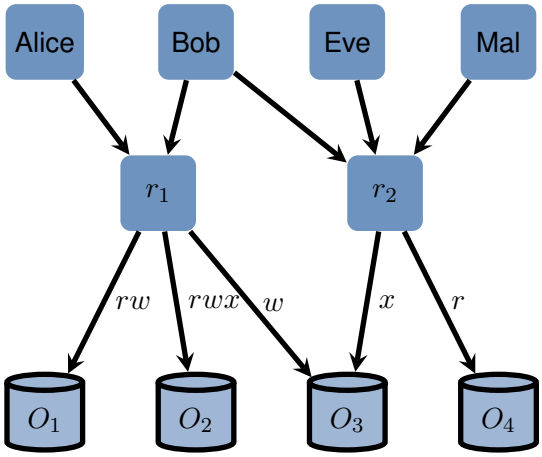


Figure 4.2: Multiple users can be assigned roles. Access rights are granted to roles only.

Users/Persons can now be assigned to roles, depending on their job description.

The next step now is to define constraints on the roles that can be assigned to a user. That is, as roles define positions of subjects, we can now identify pairs of roles that are mutually exclusive. Perhaps because the function linked with these rules generally pose a conflict of interest, or, because those roles should not be assumed at the same time (to prevent information spill between different work areas).

dynamic : not concurrent in same session by same subject

- e. g., Administrator and User
- Cashier and Guard
- CustomerAccount A and CustomerAccount B
- max. number of roles

static : not available to same subject

- e. g., President and Judge
- Manager and Employee

4.9 Location-Based Authorisation

This section introduces the concept of authorisation based on the *current* position of an entity. We utilise an example of a locking mechanism for doors that often has obvious spatial requirements before it opens.

- Based on Authentication
- Access depends on position
- Relation between person and “key”
- Example: Door Lock

A door usually is a device that provides access to a given area or passage. Often this function is coupled with the function of denying access to unauthorised persons.

Let us briefly summarise common requirements for authorisation of entry.

Authenticity Usually entry is allowed only a defined set of entities, which have to prove that they belong to that set. This is a special problem of authenticity.

Freshness It often must be assured, that a request for entry is current and not a replayed request. This is less obvious in the “real” world, as it is such a fundamental attribute of physical interaction that we rarely think about it. In electronic communication it is, nonetheless, an attribute that is not necessarily given.

Spatial Closeness You usually want that the person opening a door is the one standing close to the door. Even if you do not think about replay and masquerading attacks, you also want to prevent, that some authorised entity opens a door without overseeing who is entering that door.

Temporal Interval Some doors will only open at certain times, or at certain times for some and other times for others. Bank vaults are a common example.

Revocation Withdrawing the right of passage is a quite common operation connected with doors. If the reason why a right of passage has been granted becomes invalid, it must be possible to revoke that right.

Unlinkability It should not be possible to know, whether two different opening requests have been initiated by the same person. It has to be decided whether this requirement should only consider a passive global observer or the authorisation system itself as “attacker”. You may request various levels of privacy. Usage of pseudonyms is rather common in digital context. Unlinkability provides a stricter requirement, that is more similar to the use of keys.

4.9.1 Scenario

- person
- mobile device (smartphone)

- door lock
- authorisation/authentication server
- communication medium (local WLAN, global Internet)

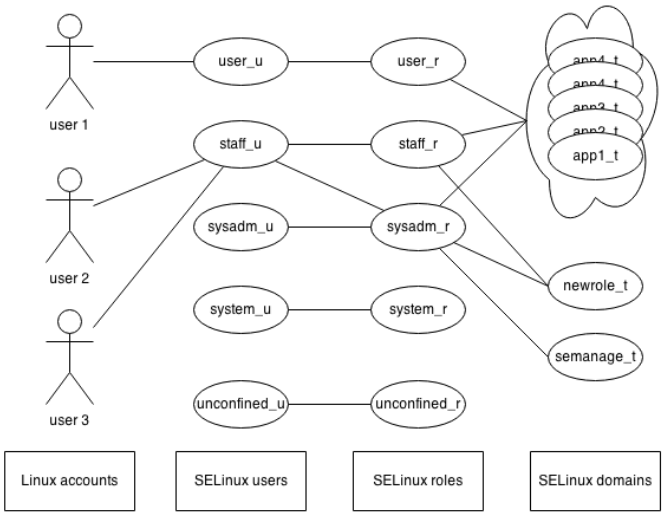
4.9.2 Solutions

- Cryptographic challenge
  - at door (e.g. QR-Code Display)
  - local number radio
  - . . . WLAN/IP Multicast
  - Polling server locally (WLAN)
- Spatial Closeness? (WLAN, changing challenge at the door?)
- Zero-Knowledge Proofs for Unlinkability/Anonymity

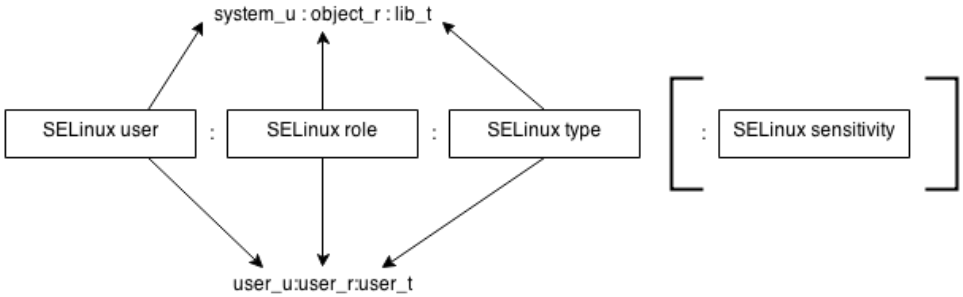
4.10 SELinux

SELinux provides an enhancement to Linux that provides Role-Based Access Control (RBAC) and MAC.

- Enhancement to Linux Kernel
- Provides
  - Message Authentication Code (MAC)
  - RBAC



[source: Gentoo Wiki]



[source: Gentoo Wiki] The context of a subject is given by its domain, defined as user, role and type. The important field for RBAC authorization is the type-field, which is checked against Security Enhanced Linux (SELinux) rules.

deny/allow ——— subject label ——— object label ——— object class ——— access right

```
allow user_t bin_t:file read;
```

- Users act only through processes (as usual)
- A Linux user account is mapped onto exactly one SELinux user
- A SELinux user is allowed one or more SELinux roles.
- A role is allowed a defined set of domains.
- Every process, i. e., subject, is, at any time, active in exactly one context.
  - Running shell in new context: `newrole -r ROLE -t TYPE`

3

Cryptology

Contents

Lernziele:

- Verständnis von Zufälligkeit?
- Cryptographische Hashes
  - Eigenschaften (unvorhersehbar, schwach/stark kollisionsresistent)
  - Birthday Paradox
- Symmetrische Verschlüsselung
- Einwegfunktionen
- Asymmetrischen Verschlüsselung
- Digitale Signaturen/Credentials

3.1 Symmetric Cryptography

Symmetric cryptography is the concept of the category of algorithms that use the same,i. e., a single key, for encryption and decryption of messages. The concept has two major fundamental problems: This means that you can never be sure, that your communication partner has the same security standards as you have. If the key is the foundation of a long-term secure communication relation, this is problematic, as with the passing of time, the possibility, that the key is somehow compromised is increased.

The second problem is, that each key is usable only for one communication channel. The result is, that you need a single (secret) key for every communication partner, that you can never share with a third person. Thus the number of all keys necessary to allow secure communication between n communication partners is n^2, i. e., quadratic growing to the number of communication partners. Also, every communication channel requires the perfectly secure exchange of the secret key before any communication can take place, or every

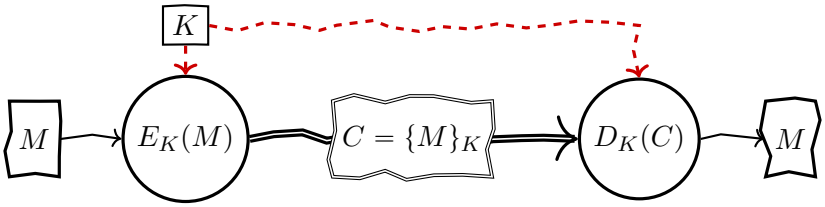


Figure 3.1: Symmetric Cipher Model

time a key is potentially compromised. In practise this turns out to be very difficult and makes the establishment of communications too complex for every area but classical high security areas.

Symmetric cryptography, also called private- or single-key cryptography, has been the only way to do cryptography (if you don't count steganography) until the 1970's. Thus, until then cryptography has been a tool only for very limited personal communication or military use.

- until 1970, the only encryption method
- Sender and Recipient share *one* key
- also:private- or single-key cryptography

Objective: hide information in transit

Plaintext M: original message

Ciphertext: scrambled message

Key K: control exact substitutions/transformations used in cipher

Encryption E: performs substitutions/transformations on plaintext

Decryption D: inverse of encryption algorithm

> Verschlüsselung ist ein Vorgang bei dem ein Datenblock derart verändert wird, dass er nur mithilfe eines Schlüssels wieder in seinen Ursprungszustand versetzt werden kann. Praktisch bedeutet dies, dass Verschlüsselung den Datenblock für alle uninterpretierbar macht, die den Schlüssel nicht kennen.

3.1.1 Requirements for Secure Symmetric Encryption

- A strong encryption algorithm
- A secret key known only to sender / receiver

Mathematical description:

- Encryption: $Y = E_k(X)$, where k is key

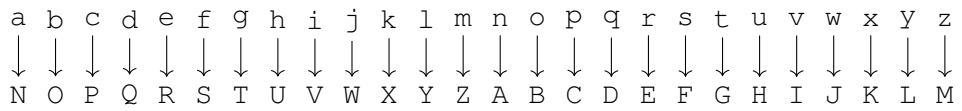


Figure 3.2: Cesar Monoalphabetic Substitution Cipher

- Decryption: $X = D_k(Y)$

Assumption:

- Encryption algorithm is known to everyone
- Hard to decrypt message given ciphertext and encryption alg.
- Therefore sufficient to keep key secret
- There is a secure channel to distribute key

3.1.2 Historic Examples

Monoalphabetic ciphers are now only found in the history section of reading books. It is simply much too easy to break such things as the classical Cesar Cipher using simple frequency counts. Nowadays all ciphers are at least polyalphabetic substitution algorithms. But much more likely they are product ciphers that utilise complex combinations of substitution and transposition.

Nonetheless, historic ciphers are a very good starting point to learn about the basic mechanisms of cryptographic algorithms. Albeit, it should be obvious, why they are called “historic”.

Monoalphabetic: Cesar Cipher

The most often used starting point and the most fundamental substitution cipher. It uses a single integer k as a key and works on every ordered alphabet (which practically is every alphabet ever used). It works by replacing each letter in a text by the k's letter in the alphabet counting from the former one. At the end of the alphabet, the count is continued from the first letter in the alphabet, which resembles the modulo-operation, that we will meet much more often in the near future.

Monoalphabetic: General Single-Letter Substitution

A slight extension to the Cesar Cipher is the generalisation uses a permutation of the alphabet as key.

I used the following simple snippet to generate a random permutation of a list of letters

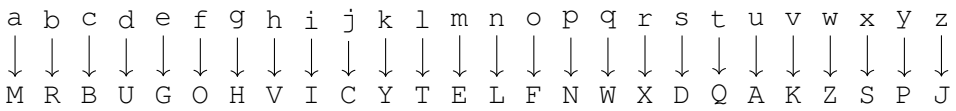


Figure 3.3: Random Monoalphabetic Substitution Cipher

```
import Data.List
import System.Random

select [] = undefined
select xs = randomRIO (0,length xs - 1) >>= (\r -> return$(\
   xs !! r, take r xs ++ drop (r+1) xs))

shuffle (out, []) = return (out,[])
shuffle (out, xs) = select xs >>= (\(r,rem) -> shuffle (r:\
   out, rem))
```

Monoalphabetic Digraph: Playfair

The Playfair Cipher increases the complexity of the classical monoalphabetic ciphers by encrypting digraphs instead of single letters. This increases the complexity of frequency analysis as the number of digraphs is approaching the number of letters squared.

The idea is to randomly arrange the complete alphabet in a box, e. g., 25 letters of the english alphabet assuming ”I” and ”J” are equal. Then the plaintext is changed into a sequence of pairs of letters. Now, for every pair find their locations in the alphabet-box and substitute following these three rules:

- if both letters are in the same row, encode each letter with its right neighbour (rolling over to the left side if necessary),
- if both letters are in the same column, encode each letter with its neighbour below (rolling over to the top if necessary),
- if both letters form the edges of a box, encode each letter with the letter in the same row, in the opposite corner of that box.

Polyalphabetic: Vigenère

The Vigenère Cipher extends the Cesar Cipher (Section 3.1.2) by using a sequence of keys, represented as *code word*, onto subsequent letters of the plaintext.

- Codeword determines encryption

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 3.4: Vigenère-Square

Transposition Ciphers

Transposition ciphers work by defining permutations to text blocks.

S-DES

S-DES is a smaller, more easy to learn variant of DES. It is, by no means to be considered as secure.

3.1.3 Ideal Block Cipher

Ideal Block Cipher

A block cipher is a cryptographic algorithm that encrypts/decrypts blocks of a given length. Formally it is a bijective map from plaintext to ciphertext.

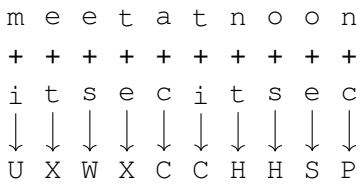


Figure 3.5: Vigenère Encryption of ”meetatnoon” using key ”itsec”

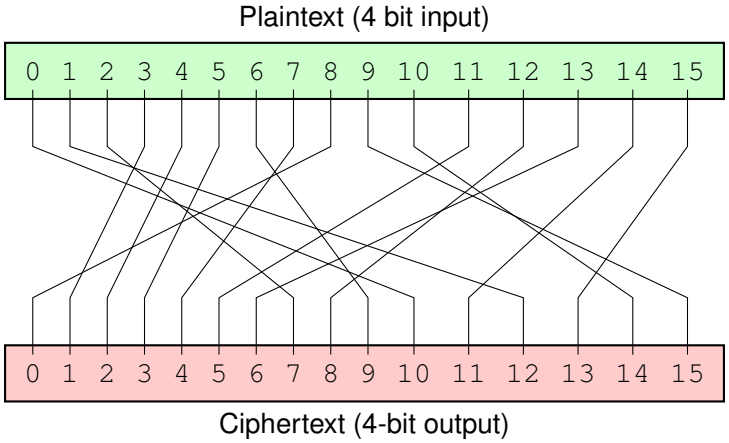


Figure 3.6: Block Cipher

	plain	cipher
	0000	1010
	0001	1100
	0010	0111
	0011	0001
	0100	0010
	0101	0011
	0110	1001
Codebook:	0111	0100
	1000	0000
	1001	1111
	1010	1110
	1011	0101
	1100	1000
	1101	0110
	1110	1011
	1111	1101

Table 3.1: Block Cipher Map

An *Ideal Block Cipher* is a permutation that maps n-bit blocks onto n-bit blocks. The key determines the permutation used for a specific cipher. The number of permutations is $2^n!$ which leaves us with a key length of $\log_2(2^n!)$ which is approximately $n \cdot 2^n$.

Issues:

- Insecure for small n, but
- impractically long keys
  - $n = 64$ (length of block in bits)
  - key length: $64 = 64 \times 2^{64} \approx 10^{21}$
  - grows exponentially

3.2 Block Cipher Modes

Encryption algorithms usually encode fixed-size blocks of data. But the size of the payload to be encrypted rarely fits into a single block. It is thus necessary to cut the data into block-sized chunks for encryption. At the destination, after the chunks have been decrypted, they can be concatenated again to reveal the original data.

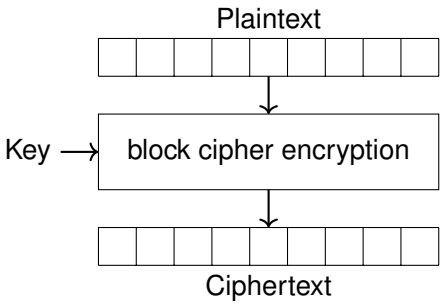
Sounds simple, but, as commonly with cryptography, the devil is in the details. In the following, we will discuss different ways to handle block-encryption to encrypt streams of data, and the shortcomings of different algorithms.

3.2.1 Electronic Code Book

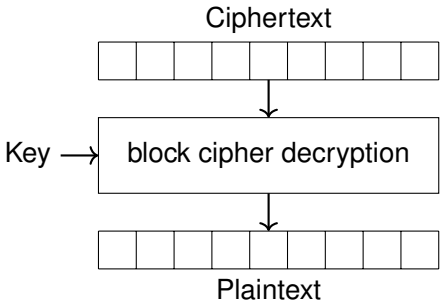
With the size of blocks, the complexity (i. e., size) of keys increases exponentially. It is thus not feasible to use a block size n that is large enough to include all possible messages or files. To encrypt data p that is larger than the cipher block size, we have to break it down into chunks p_i that are not larger than the block size. You can now apply the symmetric key k to each chunk to derive encrypted chunks c_i. This mode of operation is the Electronic Code Book (ECB) in Figure and it has the disadvantage that it fails at hiding structures within the document. (See the example in Figure 3.8)

Both operations; encryption and decryption, can be executed in parallel for each chunk. This can increase the speed in specialised hardware.

It turns out, that even something seemingly simple as applying ideal block ciphers, that have been complicated to derive, onto data can be challenging.



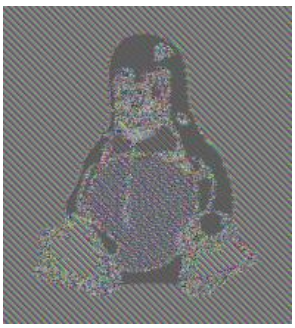
(a) Encryption



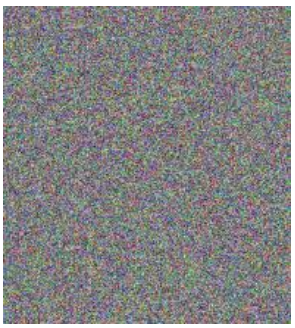
(b) Decryption

Figure 3.7: Electronic Code Book (ECB)

(a) Original



(b) ECB Encrypted



(c) Secure Mode Encrypted

Figure 3.8: Example for Electronic Code Book and Secure Encryption Mode

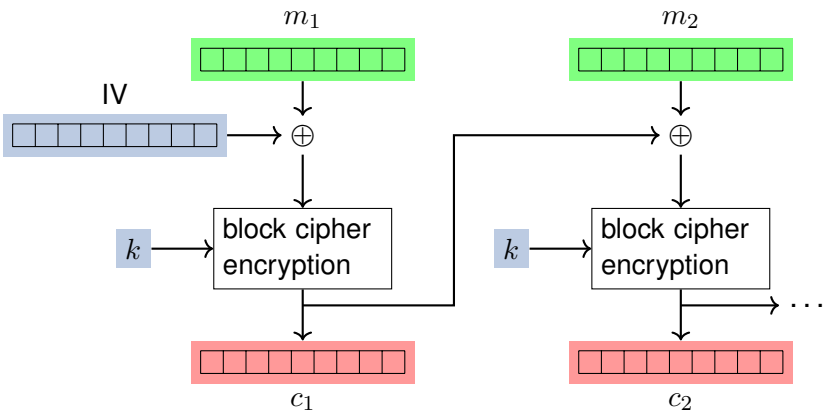


Figure 3.9: Cipher Block Chaining (CBC)

3.2.2 Cipher Block Chaining

Cipher Block Chaining (CBC) solves the problem of the ECB by creating a dependency on the previous block. In CBC mode you are able to run decryption in parallel, but encryption must happen in sequence of the chunks.

Encryption:

$$c_i = E_k(m_i \oplus c_{i-1}), i \in \{1, \ldots, r\},$$

where $c_0 = IV$ is an initialisation vector, that must be agreed upon before communication and r is the number of chunks.

Decryption:

$$m_i = D_k(c_i) \oplus c_{i-1}, i \in \{1, \ldots, r\},$$

where $c_0 = IV$ is an initialisation vector, that must be agreed upon before communication and r is the number of chunks.

The problem of CBC is the strict dependency on previous operations which results in propagation of errors. If a single bit is flipped during transmission not only the concerned block, but also the following block will be corrupted.

3.2.3 Output Feedback and Cipher Feedback

OFB and CFB are modes that can make stream ciphers from block ciphers. Both cipher modes

Output Feedback Mode

The Output Feedback Mode (OFB)

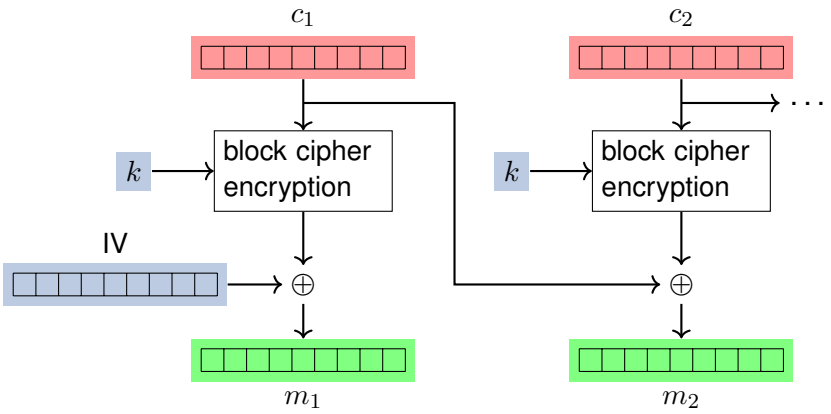


Figure 3.10: Cipher Block Chaining (CBC): Decryption

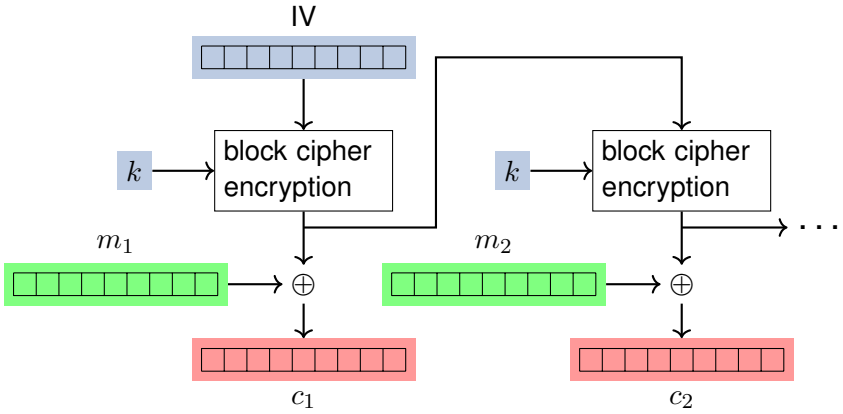


Figure 3.11: Output Feedback Mode (OFB): Encryption

$$
\begin{aligned}
c_j &= m_j \oplus o_j \\
m_j &= c_j \oplus o_j \\
o_j &= E_K(i_j) \\
i_j &= o_{j-1} \\
i_0 &= \text{IV}
\end{aligned}
$$

Cipher Feedback Mode

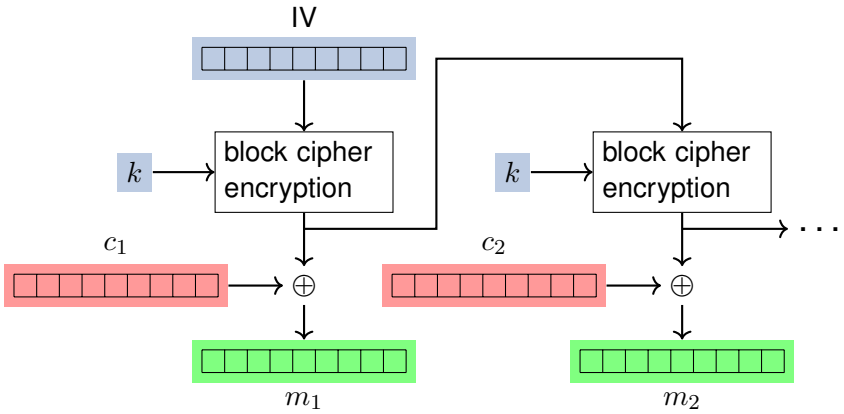


Figure 3.12: Output Feedback Mode (OFB): Decryption

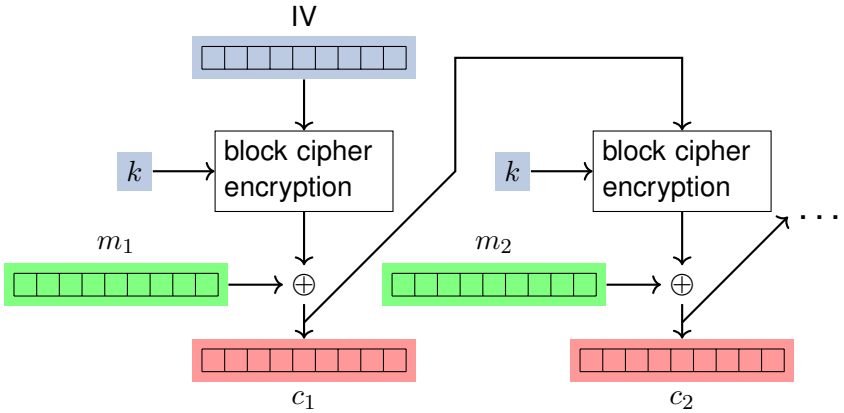


Figure 3.13: Cipher Feedback Mode (CFB): Encryption

$$\begin{aligned} c_i &= E_K(c_{i-1}) \oplus m_i \\ m_i &= E_K(c_{i-1}) \oplus c_i \\ c_0 &= \text{IV} \end{aligned}$$

3.2.4 Counter Mode

The Counter Mode (CTR) allows for parallel computation of encryption and decryption of chunks. It uses a counter to increment an individual initialisation vector for every chunk. (Figure

3.2.5 Galois/Counter Mode (GCM)

Galois/Counter Mode

`https://en.wikipedia.org/wiki/Galois/Counter_Mode`

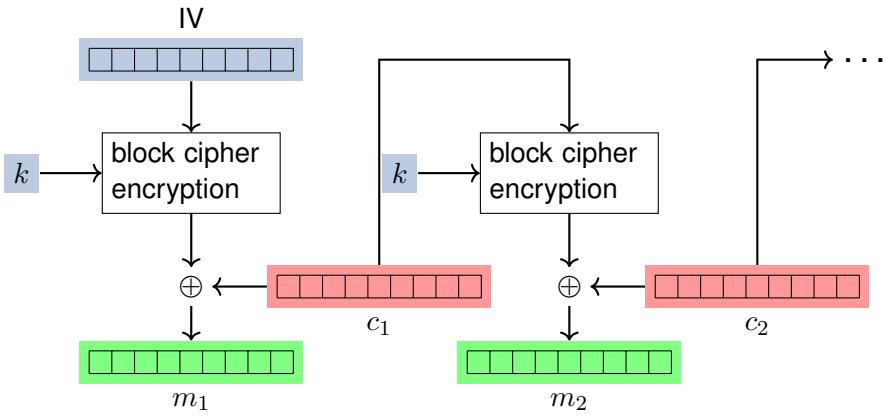
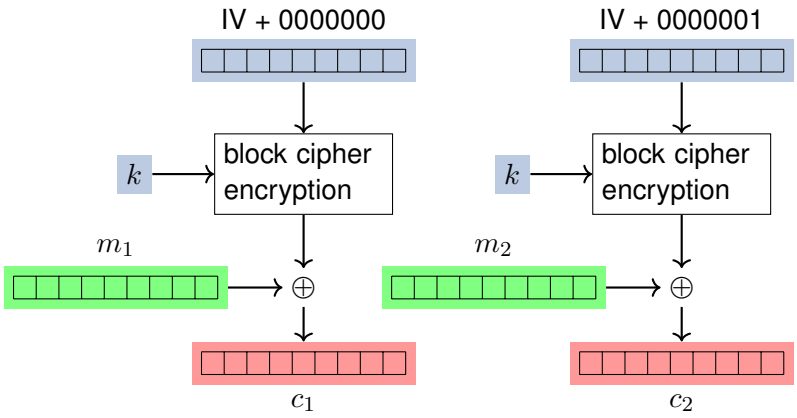
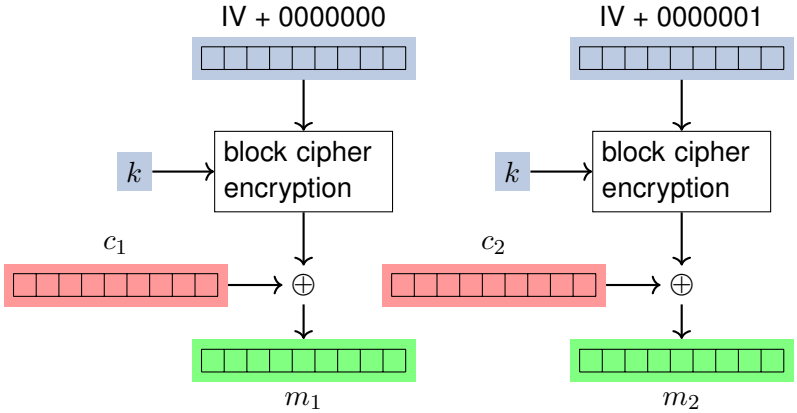


Figure 3.14: Output Feedback Mode (OFB): Decryption

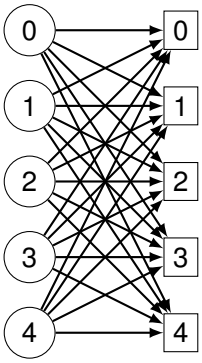


(a) Encryption

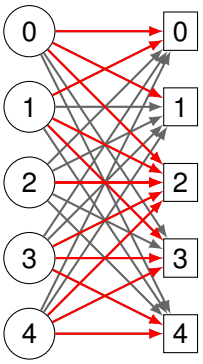


(b) Decryption

Figure 3.15: Counter Mode (CTR)



(a) Unconditional Secure Cipher



(b) Insecure Cipher

Figure 3.16: Unconditional and Insecure Cipher

3.3 Security of cryptographic algorithms

Security of computer systems is, to a large extend, based on security of cryptographic algorithms. For an algorithm to be attacked means that an unauthorized attacker is able to derive the secret key used for encryption. Generally we denote this as *breaking* the algorithm.

Thus, the big question is: how secure are the arcane algorithms that we use?

Unconditional Security

- Cipher cannot be broken, because plaintext and cipher text do not match uniquely
- No matter how much computing power or time is available

Computational Security

- Given limited computing resources (e.g. time, space) the cipher cannot be broken
- Computational cost is higher than the value of the encrypted information

Unconditional Security

Unconditional security is an attribute of a cipher, that describes that the ciphertext reveals no information whatsoever about the corresponding plaintext. A cipher that is not unconditionally secure is a cipher where an attacker can infer that certain symbols or sequences of symbols are more likely corresponding plaintext symbols than others. In Figure 3.16 this is described by the potential mappings of plaintext onto ciphertext symbols.

key:	101011
plaintext:	011001
ciphertext:	110010

Table 3.2: Example One-Time-Pad

Definition 3.3.1 (Unconditional Security)**.** The probability $P(p)$ of guessing the plain text p is equal (or at least not less) than the probability $P(p|c)$ of guessing the plain text under the condition, that the cipher text is known. A cryptographic algorithm that provides this attribute for all plain texts and cipher texts is called *unconditional secure*, formalised as:

$$\forall p \forall c : P(p|c) = P(p)$$

A cipher text that is produced by an unconditional secure encryption algorithm reveals no information whatsoever about the plain text.

One-Time-Pad

The One-Time-Pad (OTP) is a symmetric algorithm that is the only known example of an unconditional secure algorithm.

- Let $P, C, K = 0, 1^n$ be domain of p, c, k, where $|P| = |C| = |K|$
- $p, c, k0, 1^n$, key chosen randomly and uniformly distributed
- Encryption: $c = p \oplus k$, where $\oplus$ is exclusive or
- Decryption: $p = c \oplus k$

Example:

The main drawback of the OTP is that the key must have the same length as the plain- and ciphertext. The first problem is, that the key must be exchanged secure between the two communication partners before communication can take place. This is produces both overhead and restricts the usability of the scheme. Furthermore, it is very expensive to generate such large amount of random numbers. As a consequence, OTP is used mostly only by organisations that can afford such an overhead for high security applications.

Bayes' Theorem

Definition 3.3.2 (Conditional Probability)**.** The probability of an event A given event B, or A conditioned on B, is defined by Andrey Kolmogorov as the probability of the event A if the event space is limited to B:

$$P(A|B) := \frac{P(A \cap B)}{P(P)}$$

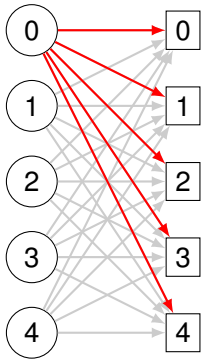


Figure 3.17: Complete Mapping of a One-Time-Pad

Theorem 3.3.3 (Bayes' Theorem)**.** *The relation between the conditional probability* $P(A|B)$ *can be derived from the probability of the reverse condition* $P(B|A)$ *as*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

To proof that a cipher is unconditionally secure it must be shown that

$$\begin{aligned}
P(p|c) =& \frac{P(p \cap c)}{P(c)} \\
=& \frac{P(c|p)P(p)}{P(c)} \\
=& \frac{P(c|p)P(p)}{\sum\limits_{p_i \in P} P(p_i)P(c|p_i)} \\
\overset{(OTP)}{=}& \frac{P(p)2^{-n}}{\sum\limits_{p_i \in P} P(p_i)2^{-n}} \\
=& \frac{P(p)}{\sum\limits_{p_i \in P} P(p_i)} \\
=& P(p)
\end{aligned}$$

3.4 Hash Functions

Please refer to [**https://nostarch.com/seriouscrypto**] for more details on hash functions and descriptions of existing hash functions.

The objective of a hash function is to map arbitrary data onto a shorter, unpredictable value in a way that cannot be reversed. The target domain is of finite

size, i. e., the length of a hash-value is limited.

Hash functions are non-injective[1] and surjective[2], because they map a large domain (that of arbitrary-length messages) onto a small domain (a fixed-width bit-field).

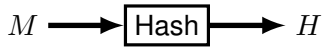


Figure 3.18: Hash function with input message M and output hash value H. (See [2])

Hash functions have to be

- unpredictable
- preimage resilient *(Einwegfunktion)*
- second-preimage resilient *(Schwache Kollisionsresistenz)*
- collision resistant *(Starke Kollisionsresistenz)*

Unpredictability means, that it is not possible, except for calculating the hash function, do get to know the resulting hash value.

```
$ echo "Hochschule Bremerhaven | Fachbereich 1" | sha512sum
680734a583a49ca0de59493ab25cf739f082f23ab21c44cdedd6b68ad0222f0ac
4b23d37ce54ac7c8a4f65bf96ec98ecd5d18845762aaad62ca26caeada1ed77  -

$ echo "Hochschule Bremerhaven | Fachbereich 2" | sha512sum
7cc3a49a63369e2bafc6698e31e3548543d3f627c9d5398e7d93ccee33e009a137
c9eb6d1a2474ba40db8098e6c5d57fbff3a689130b11d80c7eb58977e36551  -
```

Given: $h \in H$, $hash : M \rightarrow H$

Preimage: $m \in M$ with $hash(m) = h$

Required: finding m is practically impossible

Given: $m_1 \in M$, $hash : M \rightarrow H$

Second-Preimage: $m_2 \in M$ with $hash(m_1) = hash(m_2)$

Requirement:

- Preimage Resistence, and
- finding m_2 is practically impossible

[1] injective is the principle also called “one-to-one”, or $\forall a, b \in A : f(a) = f(b) \Rightarrow a = b$

[2] the principle of “onto“, or $\forall y \in Y, \exists x \in X : f(x) = y, f : X \rightarrow Y$

Collision Resistance

Given: $hash : M \rightarrow H$

Collision: $m_1, m_2 \in M$ with $hash(m_1) = hash(m_2)$

Requirement: practically impossible to find m_1 and m_2

3.4.1 Attacks on Hashes

3.4.2 Birthday Attack

How likely is a hash collision in general? Well the odds of finding a collision are surprisingly good, if only you can generate enough messages. The effect is known as the birthday paradox. The cause for its unintuitive behaviour lies in the number of pairs that can produce a collision grows quadratically.

Precisely, a collision among n messages can occur in any of $(n^2 - n)/2$ unordered pairs of messages.

Birthday Paradox

The *Birthday Paradox* describes the unexpected likelyhood of finding at least two persons within a room who where born on the same day of the year. Given:

- number of people $|N|$,
- number of days/year k,
- birthdays b

$P(b(n_i) \neq b(n_j)) = \frac{k-1}{k}$ (The argument goes as follows: you take the birthday of either of the persons as given, no matter what day it actually is. Then the probability of the other person been born on the same day is $1/365$. Likewise the probability that the second person is not born on the same day is the probability that she is born on any other of the 364 day, that is $364/365$).

If you follow above argument than, the likelyhood of three persons not being born on the same day in the year is the likelyhood of two persons times the likelyhood that the third person is not born on any of the two previous persons. For this to happen there only is a chance of $k - 2/k$, or 363 out of 365 days.

Given that $B(n)$ is the event that any of n persons has the same birthday, we can calculate the likelyhood of none of the persons having the same birthday $\neg B(n)$ as Eq 3.1

$$P(\neg B(n)) = \underbrace{\underbrace{\frac{364}{365}}_{n=2} \times \frac{363}{365}}_{n=3} \times \frac{362}{365} \times \cdots \times \frac{366-n}{365} \tag{3.1}$$

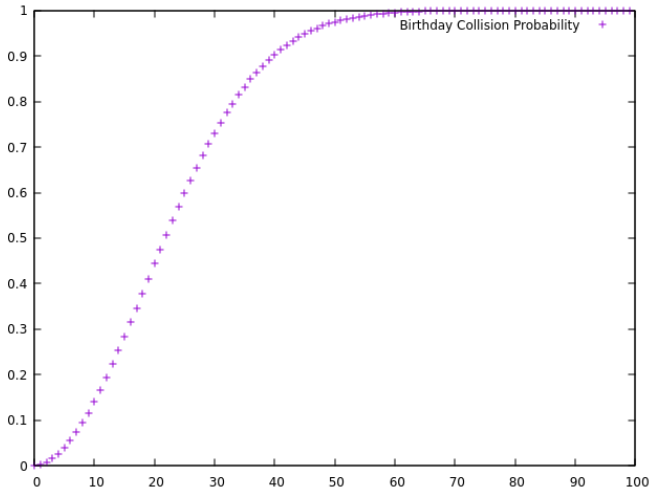


Figure 3.19: Probability of a Birthday Collision relative to the number of persons present

Let's do a quick visualisation and calculate the likelyhood of a collision occuring for n people $P(B(n))$:

```
p=1.0; for n in {1..50}; do
  echo "personen: $n; kollision: $((1-$p))"
  p=$(($p*(365-$n)/365))
done | while read a b c p;
   do echo "$p"
done | gnuplot -p -e "plot '<cat -' title \"Birthday Collision Probabili
```

Birthday Attack Algorithm

Breaking an n-bit hash is surprisingly easy. The likelyhood of having found a collision grows over 0.5 (i. e., 50%) after only $2^{k/2}$ messages.

$P(collision) > 0.5$ requires $2^{k/2}$ messages

Algorithm:

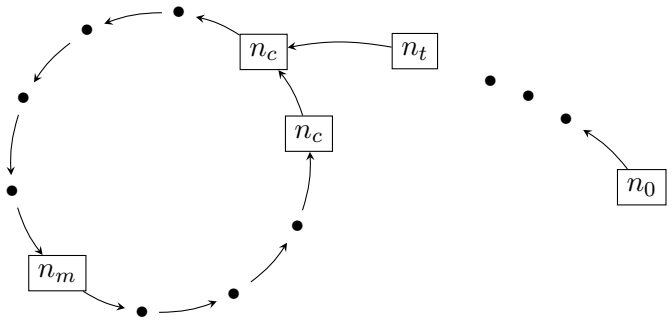
1. Generate $2^{n/2}$ random messages
2. Calculate hashes for all messages
3. Sort to have identical hash-values close

4. Find consecutive identical hashes
5. If that fails, add a new random message

While the above aproach requires a quadratic amount of memory, the Rho Method [2, p. 110] requires only a linear amount of memory. The concept is based on the idea that hash-values are essentially pseudo-random. In fact, many random number generators employ hash functions to generate random numbers, using the unpredictability of hash functions.

In brief, the Rho Methods calculates and stores a hash chain until two hash values are equal. It is based on Floyd's two fineger circle detection method.

With Floyd's two finger circle detection method you can test if a directed graph is circular. It can be used to find a hash collision in a memory-efficient manner.



There are two pointers (fingers) called hare h and tortoise t.

1. Set both pointers to node n_0
2. Every round
   - t advances one step
   - h advances two steps
3. Until $t = h$ (both point to the same node)

   , call this node "mooring point" n_m
4. Set one pointer back to n_0
5. Advance both pointers one step, until they point to the same node. Call it n_c. This node is the entry point into the circle.

If you consider a directed graph of hashes, by starting with a random hash h_0 as first node, then the nodes preceeding the entry node n_c to the circle represent a hash-collision.

- Use Floyd's two finger circle detection
  1. Pick random hash value, set $n_0 = randhash$

2. Use n_0 as starting node, setting $t_0 = h_0 = n_0$
3. Advance t by calculating $t_{i+1} := \#t_i$
4. Advance h by calculating $h_{i+1} := \#\#h_i$
5. Advance until $h_i = t_i =: n_m$ (mooring point)
6. Set $h'_0 = n_0$, $t'_0 = n_m$
7. Advance with $h'_{i+1} := \#h'_i$ and $t'_{i+1} := \#t'_i$
8. Until $h'_i = t'_i =: n_c$
9. Return h'_{i-1} and t'_{i-1} as hash collision.

3.4.3 Kerckhoffs's Principle

Auguste Kerckhoffs, a Dutch linguist and cryptographer, wrote two articles in 1883 in which he proposed a list of principles on cryptography.

1. The system must be practically, if not mathematically, *indecipherable*;
2. It must *not be required to be secret*, and it must be able to fall into the hands of the enemy without inconvenience;
3. Its key must be communicable and retainable without the help of written notes,

   and changeable or modifiable at the will of the correspondents;
4. It must be applicable to *telegraphic* correspondence;
5. It must be portable, and its usage and function must not require the concourse of several people;
6. Finally, it is necessary, given the circumstances that command its application, that

   the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

3.5 Asymmetric Cryptography

3.5.1 Basic Number Theory

This section provides some background helpful to understand the math behinde the algorithms. I mixed bits of Arithmetic, Algebra and Number Theory in order to cover the necessary basics to understand cryptographic algorithms.

Associativity $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for operation $\cdot$.

Commutativity for a set A and an operation $\cdot$ for every

$a, b \in A$: $a \cdot b = b \cdot a$.

Identity Element for a set A and an operation $\cdot$ there is an element $e \in A$, the identity element, iff

$e \cdot a = a = a \cdot e$, for all $a \in A$.

Inverse Element for a set A and an operation $\cdot$ with identity element $e \in A$, elements $a, b \in A$ are inverse elements of each other iff

$a \cdot b = e = b \cdot a$.

Divisibility "m divides n iff $m > 0$ and $n/m \in \mathbb{Z}$: Notation for "m divides n"

$m \backslash n$. (We adapt the notation from Graham, Knuth and Patashnik in [**graham1994concrete**], more commonly used is the notation $m|n$, but "|" is very ambiguously in different contexts and the backslash notation has the advantage of hinting towards m as denominator.)

Euclid's Algorithm for gcd

$$\begin{aligned} gcd(0, n) &= n \\ gcd(m, n) &= gcd(n \mod m, m), \text{ wlog.: } m \leqslant n. \end{aligned}$$

because $n \mod m = n - \lfloor \frac{n}{m} \rfloor m$

Abelian Group a set R with operation $+$ that satisfies closure, associativity, has an identity element, inverse elements and is commutative.

Finite Field (Endlicher Körper) or Galois Field[3], or $\mathbb{Z}_n$. $\mathcal{GF}$, has order p^n where p is prime and n a positive integer.

Unit an element a in $\mathbb{Z}_n$ is called *unit* if it has an inverse element $b \in \mathbb{Z}_n$, which is equivalent to $a \cdot b \equiv 1 \mod n$.

Ring an abelian group $(R, +)$ with an additional operation, usually $*$, that satisfies

$$\begin{aligned} a * (b * c) &= (a * b) * c \text{ [Associtivity]} \\ a * (b + c) &= (a * b) + (a * c) \text{ [Left Distributive]} \\ (a + b) * c &= (a * c) + (b * c) \text{ [Right Distributive].} \end{aligned}$$

If there exists an multaplicative unit e, i. e.,

$e * a = a = a * e$ then the ring is called a *unitary ring* or *ring with unity*.

[3] Évariste Galois died at the age of 20 in a duel. If ever you want to feel humbled, take a look at what this man achieved in only such a short time.

Residue Class The residue class of a modulo n:

$$[a] = \{x \in \mathbb{Z} | x \equiv a \mod n\} \tag{3.2}$$

is the set of all numbers congruent to a.

Residue Class Ring modulo n (Restklassenring) $\mathbb{Z}_n$ is the set of residue classes from $\{[a]|a\mathbb{Z}\}$, with

$\mathbb{Z}_n = \{[0], [1], \ldots, [n-1]\}$,

where $0, \ldots n-1$ are called *natural representatives*.

It is a

commutative ring with addition and multiplication $+, \cdot$, where

$[a] + [b] := [a + b]$ and $[a] \cdot [b] := [a \cdot b]$.

Prime residue class group modulo n (Einheitsgruppe) , is a multiplicative subgroup of $\mathbb{Z}_n$ consisting only of units of $\mathbb{Z}_n$, written $\mathbb{Z}_n^*$ of $\mathbb{Z}_n$. Example: $\mathbb{Z}_9^* = [1], [2], [4], [5], [7], [8]$: $1{\cdot}1 \equiv 1(mod 9)$, $2{\cdot}5 = 10 \equiv 1(mod 9)$,$4{\cdot}7 = 28 \equiv 1(mod 9)$, $8 \cdot 8 = 64 \equiv 1(mod 9)$

For prime residue class groups $\mathbb{Z}_p^*$ each representative $a \in \{1, \ldots, p-1\}$, excluding 0, corresponds to a unit.

Generator an element $a \in M = \{0, \ldots, q-1\}$ is called *generator* if there is, for all $m \in M$ a $p \in M$ with $m \equiv a^p \mod q$.

Primitive root A generator $g \in \mathbb{Z}_n^*$ is called *primitive root*.

Cyclic Group Let G be a finite group. G is cyclic, if there is a $g \in G$ which generates G, i. e., $G = \{g^1, g^2, \ldots, g^{ord(g)-1}, g^{ord(g)} = e\}$, where e is the identity element of G

- g is called generator of G.
- Example: Let $G = \mathbb{Z}_5^*, [2] \in \mathbb{Z}_5^*$, then $ord([2]) = 4 \in \mathbb{Z}_5^*$. $[2]$ is generator of $\mathbb{Z}_5^*$, because $\mathbb{Z}_5^* = \{[2], [2]^2, [2]^3, [2]^4\} = \{[2], [4], [3], [1]\}$

If p is prime, then $\mathbb{Z}_p^*$ is cyclic and the number of generators is $\phi(p-1)$.

Euler's Totient Function $\phi(n)$ is the number of elements in $\mathbb{Z}_n$ that are *relative prime* to n, i.e. $gcd(x, n) = 1$. Especially it is multiplicative, meaning that if m, n are relative prime to each other, then

$$\phi(mn) = \phi(m)\phi(n).$$

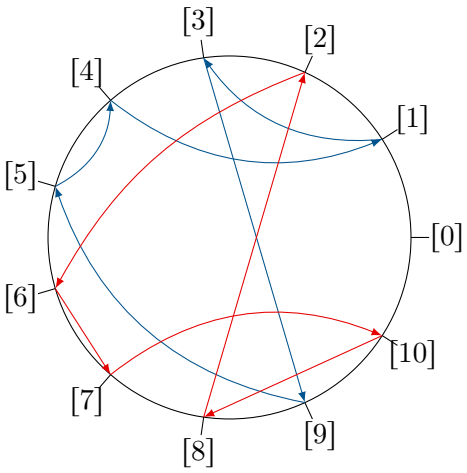


Figure 3.20: Cyclic Groups $\mathbb{Z}_{11}$, arrows show generating funktion "multiplication by three"r

Let's have a prime decomposition $n = p_1^{e_1} \cdot \cdots \cdot p_r^{e_r}$, then

$$\begin{aligned} \phi(n) &= \phi(p_1^{e_1}) \cdot \cdots \cdot \phi(p_r^{e_r}) \\ &= p_1^{e_1}(1 - \frac{1}{p_1}) \cdot \cdots \cdot p_r^{e_r}(1 - \frac{1}{p_r}) \\ &= p_1 \cdot \cdots \cdot p_r(1 - \frac{1}{p_1}) \cdot \cdots \cdot (1 - \frac{1}{p_r}) \end{aligned} \tag{3.3}$$

Euler's Theorem if $gcd(a, n) = 1$ then $a^{\phi(n)} \equiv 1 \mod n$

Fermat's Little Theorem p is prime, then for all $a \in M = \{1, \ldots, p-1\}$: $a^{p-1} \equiv 1 \mod p$.

Efficient Modular Arithmetic

Modular Arithmetic:

Calculations in $\mathcal{GF}(23)$:

$$\begin{aligned} 5^2 &\equiv 5 \times 5 \equiv 2 \\ 5^4 &\equiv 5^2 \times 5^2 \equiv 4 \\ 5^8 &\equiv 5^4 \times 5^4 \equiv 16 \\ 5^{16} &\equiv 5^8 \times 5^8 \equiv 3 \\ 5^{23} &\equiv 5^{16} \times 5^7 \equiv 5 \mod 23 \end{aligned}$$

In 1976 two cryptographers; Whitfield Diffie and Martin Hellman, rocked the world with the idea to revolutionise cryptography. Their requirements where

quite simple: get rid of the excruciating need for secure key distribution, which almost ridiculed the idea of secure message exchange and provide a cryptographic system that would allow for signatures.

- Diffie, W. & Hellman, M. New directions in cryptography, 1976:
  - "minimize the need for secure key distribution"
  - "supply equivalent of a written signature"

"A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}$ and $\{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations,

$$E_K : \{M\} \longrightarrow \{M\} \tag{3.4}$$
$$D_K : \{M\} \longrightarrow \{M\} \tag{3.5}$$

on a finite message space M, such that

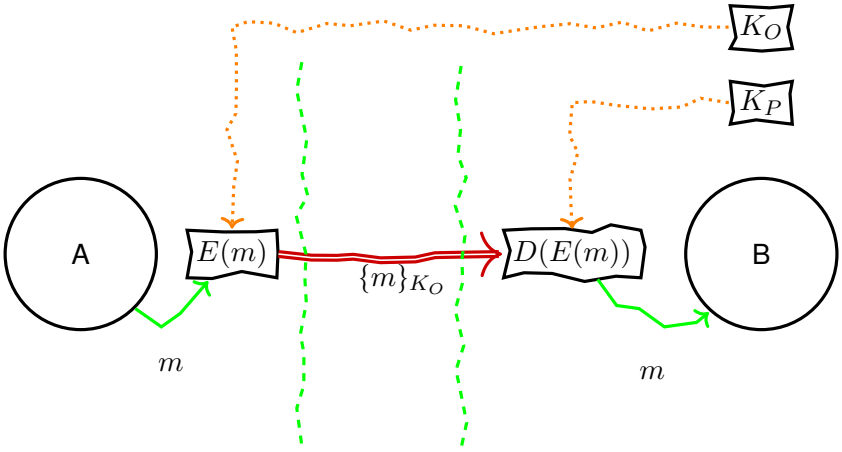
1) for every $K \in \{K\}$: E_K is the inverse of D_K,
2) for every $K \in \{K\}$ and $\{M \in \{M\}$, the algorithms E_K and D_K are easy to compute,
3) for almost every $K \in \{K\}$, each easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K,
4) for every $K \in \{K\}$, it is feasible to compute inverse pairs E_K and D_K from K."[**DH76Newdirectionsin**]

They further introduce an algorithm for exchange of keys that can be done on a public channel. The Diffie-Hellman Key Exchange is using calculations over finite fields, which is, why we have to start with some basic number theory first.

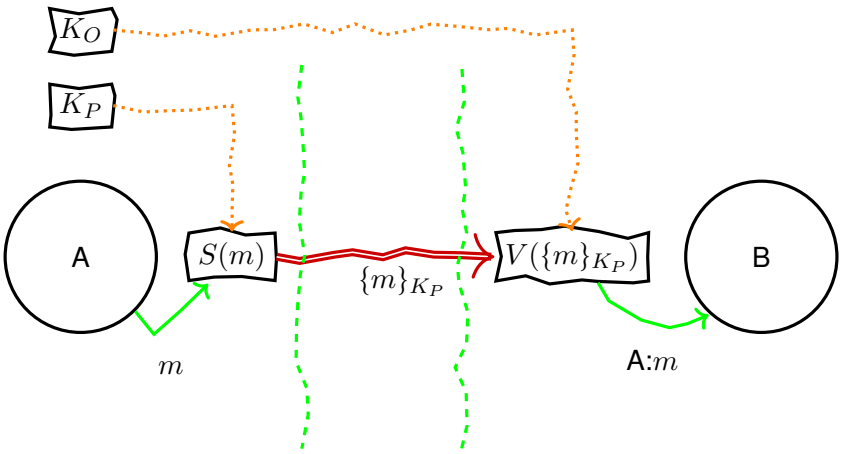
Asymmetric cryptography makes encryption into a one-way-road: If you apply the public key, only the holder of the private key can reverse, i. e., decrypt, the message again. The one encrypting the message itself is not able to reverse the operation. But generally, this is not a hindrance, as asymmetric encryption is rarely applied directly to a message. (See)

Hybrid Encrypton

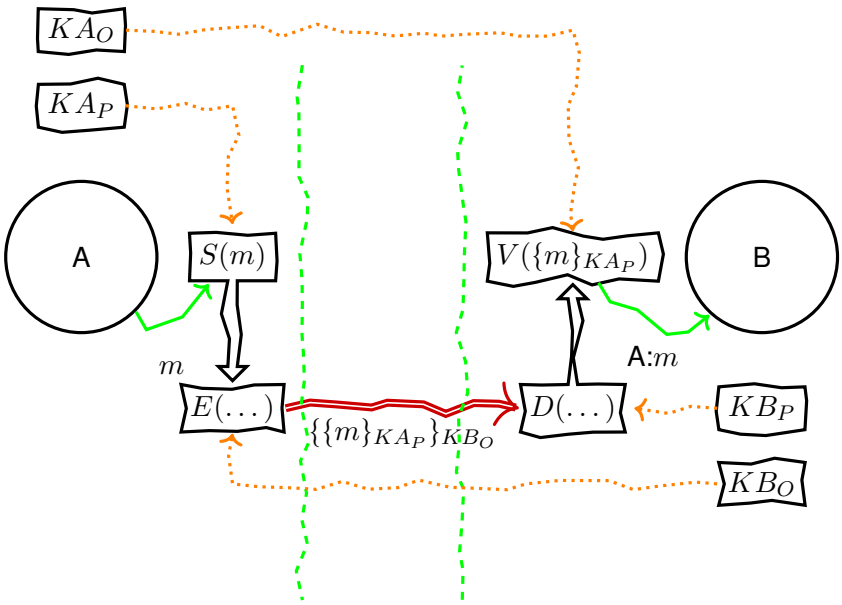
To use an asymmetric encryption algorithm, Bob has to prepare by creating a pair of related keys; one public and one private keys. The public keys he has to distribute to all principals he wants to receive encrypted messages from. If Alice wants to send an encrypted message to Bob, she has to know the correct, i. e., authentic, public key of Bob. She then can apply Bob's public key to a message, using the trapdoor function that is the asymmetric encryption algorithm, and send the encrypted message to Bob. Bob then can apply his matching private key and retrieve the original message.



Asymmetric cryptography provides a completely new application for cryptographic algorithms as compared to symmetric cryptography. Using a private key is an operation only the holder of that key can facilitate. This allows, for example, to interpret the result of that operation as a proof that the holder of that private key has seen the message and approves it. The most common application of this is to provide authentication of the source of a message, or as it is commonly refered to, a *signature*.



On the other hand, asymmetric cryptography can be used for the original aim of cryptography, to keep the contents of messages a secret to all but the holder of a secret key. In general, it seems useful to combine both functions, encryption and signature.



The general, yet not mandatory, practise is to first sign a message and then encrypt both the message and the signature. In that way no information on the sender of an encypted message is leaked. The reason to do it in that order is, that oftentimes a signature is interpreted as agreement to the signed statement. This assumption can be made as long as the plaintext message is signed and not the encrypted message. This is denoted by the fifth principle in the “Prudent Engineering Practice for Cryptographic Protocols” by Needham and Abadi.

> Principle 5: When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of the message. [**abadi96prudent**]

3.5.2 Diffie-Hellman Key Exchange

Exchanging secrets has always been a difficult problem. For sure, you could meet “in private” and have a conversation. But this is very costly, as it requires to meet physically at the same space and time, and further ensure that no other is eavesdropping. In practise secret messages have to be transported over channels that do not fullfill the requirements of a private meeting. The message always has to be entrusted to a transport that is not the intended recipient. And, by “entrusted” we explicitly mean, that sender or recipient have no way to ensure that the transport is not peeking.

Symmetric encryption has been a huge step forward, as it allows to apply an algorithm, via a secret key, to the message which makes it intelligible unless you are able to reverse that algorithm, via the same secret key. And, while this

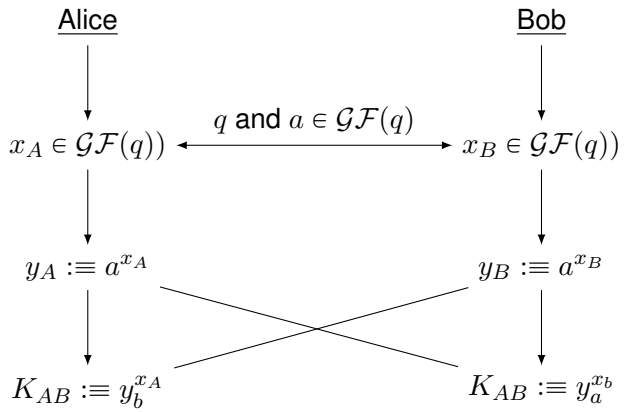


Figure 3.21: Diffie-Hellman Key Exchange (for two parties), all $\equiv$ are $\mod q$

generally enables you to exchange secrets over untrusted channels, you now have a secret key that you need to exchange in secret before you can communicate secretly. An obvious chicken and egg situation, that is only relieved because you can do that exchange when you have an occassion to confer in private, as long as it happens beforehand.

We obviously gained a lot by encryption algorithms. And until Diffie and Hellman posed their requirements for asymmetric encryption algorithms very few have objected against the seemingly fundamental constraints of encryption. It seems that it helps to think about the seemingly impossible. Because shortly after Diffie and Hellman posed their white-paper on asymmetric encryption and requirements therefor, the provided a first solution.

The Diffie-Hellman Key Exchange is a distributed algorithm (or a communication protocol) that allows two (or more) parties to cooperatively generate a shared value over a public channel without disclosing this value. Thus, after the exchange, both parties know the same secret while no observer can know which value that is, although the observer has seen all exchanged messages.

Sounds like magic to you? You are not the first person to feel that way, but in the end it is all some fine but explainable mathematics.

Diffie-Hellman Key Exchange is secure as long as it is hard computationally hard to reverse the power-function in residue class rings. Later on this assumption has become known as the *Strong RSA Assumption*.

Strong RSA Assumption Given an RSA modulus n and an element $u \in \mathbb{Z}_n^*$ it is hard to find $e > 1$ and v such that $v^e \equiv u \mod n$.

The Diffie-Hellman Key Exchange is secure under one assumption. That is, the communication between the two parties is not intercepted by a Person-in-the-Middle (formerly known as Man-in-the-Middle). If an attacker is able to

intercept the communication and pose (spoof) as Alice or Bob to the respectiv other side, it is easy to have an individual DH Key-Exchange with both sides. In that way we end with the Person-in-the-middle sharing a secret with both sides, leaving Alice and Bob with the wrong assumption of their respective secret being shared only by them.

But, the attacker still is not able to partake in the shared secret of Alice and Bob. If Alice and Bob use their secrets to encrypt their further communication, the attacker will be able to read all messages, but Alice and Bob are not able to decrypt the messages send by the other because they are using different keys. That means the Person-in-the-middle will have to intercept and re-encrypt every and all messages exchanged between Alice and Bob using the secrets they falsely assume to be shared with each other. Alice or Bob may grow suspicious if they start receiving messages from each other that they cannot decrypt with the assumed shared secret.

3.5.3 RSA

RSA, named after his inventors Ron Rivest, Adi Shamir and Leonard Adleman, is one of the first and most commonly known and used algorithms for asymmetric cryptography. Much more important, the principles used are easy to understand and used or comparable to the underlying concepts of more recent algorithms.

You should take a glimpse at the section on number theory and algebra to polish your math.

- Popular asymmetric algorithm
- Based on discrete logarithm
- Vulnerable to Davida's Attack, if key pairs for signing and encryption are the same.

Key Generation

1) Choose two prime numbers p and q at random (stochastically independent) with $|p| \approx |q| = l, p \neq q$
2) Calculate $n := pq$
3) Choose c with $3 \leqslant c < (p-1)(q-1)$ and $gcd(c,(p-1)(q-1)) = 1$, remember $\phi(n) = (p-1)(q-1)$
4) Calculate d using p, q, c as multiplicative inverse of c in residue class group $\mathbb{Z}/n\mathbb{Z}$ so that: $c \cdot d \equiv 1 \mod \phi(n)$
5) Publish c and n as public keys.

Encryption

En-/decryption exponentiation with c respectively d in $\mathbb{Z}/n\mathbb{Z}$

Proposition: $m \in \mathbb{Z}/n\mathbb{Z} : (m^c)^d \equiv m^{c \cdot d} \equiv (m^d)^c \equiv m \mod n$

Signature

Exponentiation with d respectively c in $\mathbb{Z}/n\mathbb{Z}$

Proposition: $m \in \mathbb{Z}/n\mathbb{Z} : (m^d)^c \equiv m^{d \cdot c} \equiv (m^c)^d \equiv m \mod n$

Correctness Proof

Proposition: $m \in \mathbb{Z}/n\mathbb{Z} : (m^c)^d \equiv m^{c \cdot d} \equiv (m^d)^c \equiv m(mod\, n)$

Remember:

$$cd \equiv 1(mod\, \phi(n)) \Leftrightarrow \exists k \in \mathbb{Z} : cd = k\phi(n) + 1$$

Therefore

$$m^{c \cdot d} \equiv m^{k \cdot \phi(n)+1} \equiv m(mod\, n),$$

by Euler's Theorem: $m^{\phi(n)} \equiv m(mod\, n)$.

It follows for all m coprime to p, i. e., $gcd(m, p) = 1$ (Fermat's Theorem):

$$m^{p-1} \equiv 1(mod\, p)$$

Because $(p-1)|\phi(n)$: $m^{k \cdot \phi(n)+1} \equiv m^{k \cdot (p-1)(q-1)+1} \equiv m \cdot \underbrace{\underbrace{(m^{p-1})}_{=1}^{k \cdot (q-1)}}_{=1} \equiv m.$

Because $\phi(n)$ is, by construction of the finite field of the exponent, equal to $(p-1)(q-1)$, with p, qprime, that means $m^{k \cdot \phi(n)+1}$ can be expressed as $m^{k \cdot (p-1)(q-1)+1}$. Now, we can reformulate this into $m \cdot (m^{p-1})^{k \cdot (q-1)}$ and because of Fermat's Little Theorem conclude that m^{p-1} equals 1 (in $\mathbb{Z}/n\mathbb{Z}$ and also $(m^{p-1})^{k \cdot (q-1)}$. This renders $m^{c \cdot d} \equiv m(mod\, n)$.

Security of RSA

RSA is not only easy to understand, but has been heavily attacked by cryptoanalysts. In 1991 the RSA Laboratories funded a challenge for factoring large integers in order to crack RSA keys. The challenge ran until 2007 but the milestones of the challenge are still used as measurement of RSA security. (Table 3.3)

Number	Bits	Cash prize	Factored on	Factored by
RSA-100	330	$1,000	Apr 1, 1991	Arjen K. Lenstra
RSA-110	364	$4,429	Apr 14, 1992	Arjen K. Lenstra and M.S. Manasse
RSA-120	397	$5,898	Jul 9, 1993	T. Denny et al.
RSA-129	426	$100	Apr 26, 1994	Arjen K. Lenstra et al.
RSA-130	430	$14,527	Apr 10, 1996	Arjen K. Lenstra et al.
RSA-140	463	$17,226	Feb 2, 1999	Herman te Riele et al.
RSA-150	496		Apr 16, 2004	Kazumaro Aoki et al.
RSA-155	512	$9,383	Aug 22, 1999	Herman te Riele et al.
RSA-160	530		Apr 1, 2003	Jens Franke et al., University of Bonn
RSA-170	563		Dec 29, 2009	D. Bonenberger and M. Krone
RSA-576	576	$10,000	Dec 3, 2003	Jens Franke et al., University of Bonn
RSA-180	596		May 8, 2010	S. A. Danilov and I. A. Popovyan, Moscow State
RSA-190	629		Nov 8, 2010	A. Timofeev and I. A. Popovyan
RSA-640	640	$20,000	Nov 2, 2005	Jens Franke et al., University of Bonn
RSA-200	663		May 9, 2005	Jens Franke et al., University of Bonn
RSA-210	696		Sep 26, 2013	Ryan Propper
RSA-704	704	$30,000	Jul 2, 2012	Shi Bai, Emmanuel Thomé and Paul Zimmerma
RSA-768	768	$50,000	Dec 12, 2009	Thorsten Kleinjung et al.
RSA-896	896	$75,000		
RSA-1024	1024	$100,000		
RSA-1536	1536	$150,000		
RSA-2048	2048	$200,000		

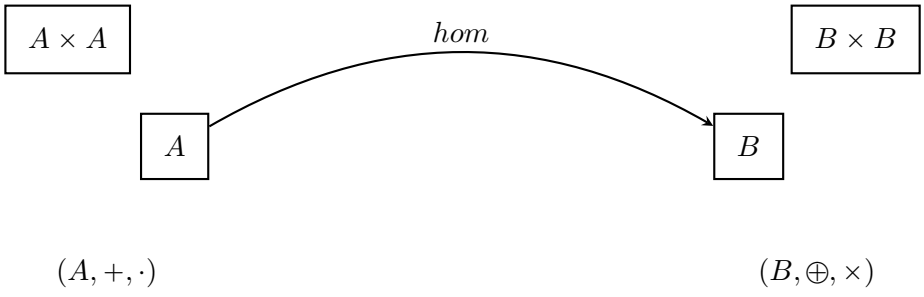
Table 3.3: RSA Challenge Milestones [`https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`]

In 2001 Dan Bernstein proposed a technique that could increase the factoring success of existing algorithms by a factor of 3. This work has been criticised by Arjen K. Lenstra, Adi Shamir, Jim Tomlinson, and Eran Tromer as being to optimistic. Instead they argue that the effect of Bernsteins number field sieves would be only improve algorithms by a factor of 1.17.

RSA is a partially homomorphic encryption algorithm. This remarkable feature provides that certain operations can be executed on *encrypted* data, without decrypting the data beforehand. If the algorithm is homomorphic, than this operation is equivalent to an operation on the decrypted data. This, homomorphic encryption algorithms allow mathematical operations on data, without being able to decipher the data[4]

A homomorphism is a mapping between the elements of two algebraic structures of the same typ. Meaning both structures provide a similar set of operations (with same arity) on their respective sets.

Sets A, B and operations $+, \cdot, \oplus, \times$.



e. g., $\forall x, y \in A : hom(a) \times hom(b) = hom(a \cdot b)$

Homomorphisms can be understood as maps that retain the essential structure of the preimage within the target domain. If an inverse homomorphism $hom^- 1$ exists, both structures are called *isomorph* and the homomorphism is called an *isomorphism*.

The homomorphism in RSA allows for homomorphic multiplication. Thus, you could multiply on encrypted data. The only requirement is, that the data be encrypted with the same RSA-key pair.

$$E(x) \cdot E(y) = E(x \cdot y) \tag{3.6}$$

[4] pretty magical, ain't it?

Now, homomorphic encryption can be used to delegate computations on sensible data, without disclosing that data.

In [**davida1982chosen**] utilises the RSA homomorphism to manipulate a principal into signing an encrypted message. If the same public/private pair of keys is used for both encryption and signature, then the application of the signature reverses the encryption. In order to hide this from the signer, the encrypted message is "blinded" beforehand.

Davida's attack not only teaches us a conceptual vulnerability of asymmetric cryptography but also the very useful technique of blinding, which can also be used for blind signatures in electronic money schemes and anonymous credentials.

In the first step the attacker intercepts a message m encrypted with the public key e of the victim. The attacker then multiplies an arbitrary value y encrypted for the victim.

$$
\begin{aligned}
\text{given: } & m^e & & \\
\text{select: } & y & \longrightarrow \quad & x = y^e \\
\text{blind: } & m^e & \longrightarrow \quad & x \cdot m^e \\
\text{victim signs: } & x \cdot m^e & \longrightarrow \quad & (x \cdot m^e)^d \\
& (c^{\cdot e})^d & \overset{\text{RSA Hom.}}{\Longrightarrow} \quad & x^d \cdot m^{e \cdot d} \\
& & & = y^{e \cdot d} \cdot m^{e \cdot d} \\
& & & = y \cdot m \\
\text{unblind: } & y \cdot m & \overset{/y}{\Longrightarrow} \quad & m
\end{aligned}
$$

The combined value of encrypted message and value is the forwarded and the victim is foiled into signing the value, e. g., by using it as part of another statement. By the RSA homomorphism this is the same as applying the private key individually to the encrypted message and the encrypted, attached value.

The result is the product of attached value and message. As the value has been created by the attacker, he is able to extract the unencryptey message by dividing through y.

This provides us with two general rules on the handling of asymmetric keys:

1. Never use the same key-pair for encryption and signature, and
2. Be clear about what the meaning of a signature of a given document is.
3. Do not sign messages that you cannot read/understand.

3.6 Message Authentication Code (MAC)

A Message Authentication Code (MAC) is data added to a message to prove the authenticity of that message. MAC are produced using a cryptographic operation with a secret key with a corresponding operation to verify that the MAC is derived from the concerned message and the secret key.

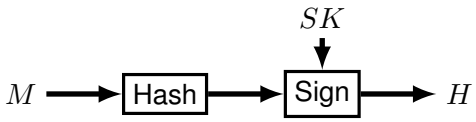


Figure 3.22: Secure (signed) Hash function with input message M and output signed hash value S, authenticated by application (sign) of a secret key SK (See [2])

Message authentication is concerned with:

- protecting the integrity of a digital message, i. e., unintended alteration of message is detectable
- validating identity of originator, which is, often, a necessary prerequisite to integrity protection. Or, the other way round, integrity of originator statements is a form of authenticity
- Sometimes: Timely and correct sequence of message, which can

The fundamental technique for MAC is to mix a given message with a secret and then perform a hashing operation on the result. The two most simple forms generate MAC from (unkeyed) hash functions

Prefix or Suffix Construction of MAC Secret-Prefix

$$\mathcal{H}(K|M)$$

- Length-Extension Attack works for some hash functions, e. g., SHA-2 family. It allows to compute

  $$\mathcal{H}(K|M_1|M_2)$$

  for unknown key K and message M_1.
- Variable Key-Length Attack if the length of keys is variable, then one could have equal hash-values for different messages, if the start of a message is equal to the remainder of the shorter key. Thus you can have two different messages with the same hash. That never is a good thing to have.

Secret-Suffix

$$\mathcal{H}(M|K)$$

- Exploit Hash-Collision of Messages

Hash-based Message Authentication Code (HMAC)

Hash-based Message Authentication Code (HMAC) has superior security against message extension attacks and thus is probably the most widely used concept for MAC. IPsec and TLS (Section **??**) are just two examples for usage. It is described in RFC 2104 and RFC 6151. You find the description of the algorithm below and depicted in Figure 3.23.

Define two fixed and different strings $ipad$ and $opad$ (the 'i' and 'o' are mnemonics for inner and outer):

$$\begin{aligned} ipad &= \text{the byte } \texttt{0x36} \text{ repeated } B \text{ times} \\ opad &= \text{the byte } \texttt{0x5C} \text{ repeated } B \text{ times.} \end{aligned}$$

To compute HMAC for the data 'text':

$$H(KXORopad, H(KXORipad, \text{ text})).$$

Namely,

(1) append zeros to the end of K to create a B byte string (e.g., if K is of length 20 bytes and $B = 64$, then K will be appended with 44 zero bytes 0x00)

(2) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with $ipad$

(3) append the stream of data 'text' to the B byte string resulting from step (2)

(4) apply H to the stream generated in step (3)

(5) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with $opad$

(6) append the H result from step (4) to the B byte string resulting from step (5)

(7) apply H to the stream generated in step (6) and output the result

[RFC 2104]

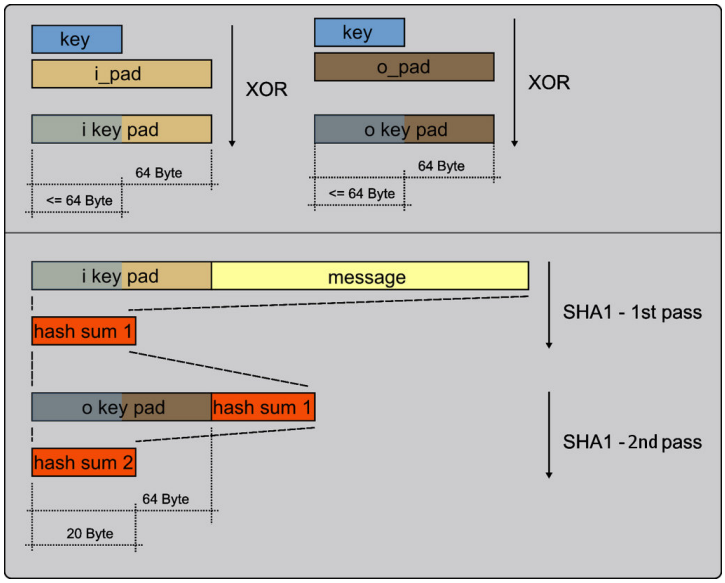


Figure 3.23: HMAC

4

Authentication

Contents

Authentication almost never is the actual objective but only the technical means to something else. Most often the objective is authorization (See Chapter **??**).

We have learned, that it is difficult to establish secure connections without authentication of at least one communication partner. Authenticity is prerequisite to most security measures.

Secrecy based on cryptographic keys requires authentic key material. If the source of a key is not securely known, a sender could not distinguish whether he is encrypting for the intended destination or an attacker.

Access control becomes useless, if a system cannot distinguish between different subjects, let alone cannot prevent a subject to pose as a different subject. Even protection of availability often requires to distinguish between acceptable users and illegitime users.

Self-Sovereign Identity, kurz: SSI

https://toolkit-digitalisierung.de/digital-identity/

Authentication in general is the "act of confirming the truth of an attribute of a datum or entity."[1]

In this chapter we introduce techniques and structures to create and verify that a given entity is using an authentic identity. In general, three distinct classes of methods are distinguished:

Knowledge something one knows

Possession something one holds

Measurement something one is (i. e., Biometry)and which can be measured in any way.

It is possible (and more secure mostly) to combine multiple classes together. For example authentication of a human being by identity card uses possession (the paper/plastic document) and measurement (the picture on the card and probably the signature). Another form of combination can be used for au-

[1] https://en.wikipedia.org/wiki/Authentication, 2013-12-06

thentication of machines towards a human, for example the form of casing, the “knowledge” of certain data, the location. (Find some illustrations in the slides.)

Authentication can happen between different types of entities, humans against humans, humans against machine, machines against machines and machines against humans.

Figure 4.1: Machine-to-Human (M2H) Authentication

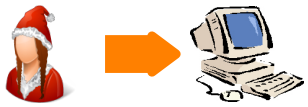


Figure 4.2: Human-to-Machine (H2M) Authentication

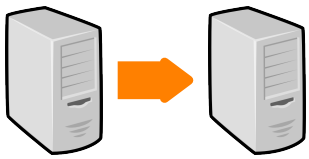


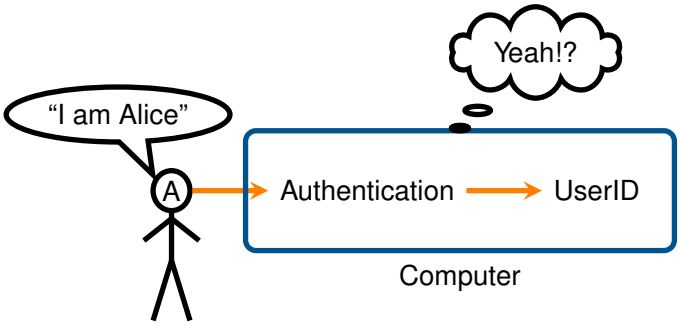
Figure 4.3: Machine-to-Machine (M2M) Authentication

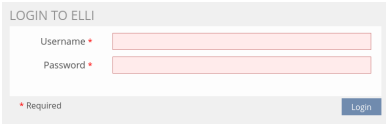
The first section introduces techniques to authenticate entities, especially human entities.

4.1 Password Authentication

Passwords are the most common method for authentication of humans. Passwords are also both a very simple and very vulnerable method for authentication. The *password authentication method* is based on knowledge of a string of letters, numbers and punctuation that is used as a static response given by a human to verify his claim on a given identity.

Passwords are probably the most common and most despised method for authentication. Users commonly misunderstand the importance of sticking to the guidelines of the experts thus leaving their accounts wide open. Experts mostly fail to create better and less static methods of authentication by knowledge. Generating and memorising passwords is a excruciating pain for all.





Username:

- Identifies user within system
- (communication identifier)

Password:

- Prove identity statement correct
- Not guessable!

4.1.1 Password Vulnerabilities

Passwords are probably the worst way to provide secure authentication. Mostly because the security of the password method depends mostly on user behaviour. Passwords are vulnerable

An (incomplete) list of vulnerabilities:

- Password guessing (brute force)
- Password guessing (personal context) See the movie “Wargames” for a neat example.
- Popular password attack

- Exploiting multiple password use every good password-cracker worth his/her salt[2] will have a pipeline deployed that checks every uname/password-combination found with popular services.
- Evil Maid Attack:
  - Workstation hijacking
  - Electronic monitoring (key-logger, van-Eck)
- Scrape it from the memory (mimikatz)
- Offline
  - Wordlist enumeration
  - Modification Rules
  - Hash Calculation (optimized)
  - Rainbowtables

4.1.2 Password Countermeasures

First thing in defence always is to identify the interfaces to the assets. From that there are a few general principles to adhere to.

Interfaces:

- user-“memory”
- authentication process
- password database

4.1.3 Login Process

Login:

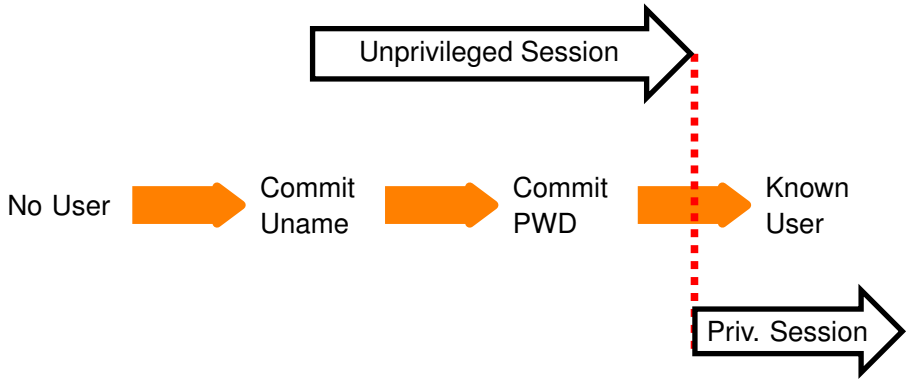
- Encrypted network links
- Exponential Waiting-Time
- Random-Session-ID
- Pinning-Prevention: separate the “public” unauthenticated session from the authenticated session.

Session:

- Account lockout mechanisms

[2] pun intended

- automatic
- effective(!): there are examples where
- usable



Remember that authentication rarely is the objective, but a technique to achieve something. In that respect an identity authentication process, e. g., a login process, is a transition from an unprivileged state into a state with higher privileges. Or, in simpler terms via login a user gains entry into a system.

There are a few guidelines for login processes (or identity authentication processes in genereral):

- the unprivileged session must be protected from eavesdropping
- session identifiers have to be randomly re-assigned during the privilege transition
- the authentication service must itself be authenticated during the unprivileged session already

4.1.4 Password Storage

- Never(!) store plaintext
- Stop unauthorized access to password file
- Intrusion detection measures

4.1.5 User-side Password Handling

- Hard to guess passwords
- Unique Passwords (1/Account)

A crucial part of the security of passwords is handling of the passwords by the user. Users have to choose secure passwords and store them in a way that the

clear text is accessible to them. There are essentially two ways for a user to handle passwords:

- Random Passwords and Secure Storage
- Manual Algorithms that are easy to remember

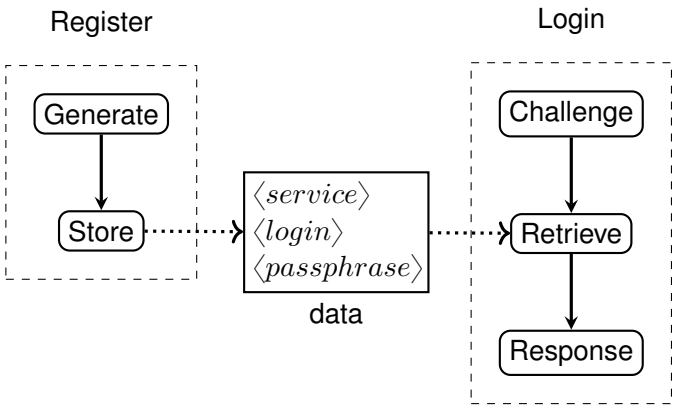


Figure 4.4: Password Data Usage

Passphrases are only one part of the data required in order to login to a service. In order to retrieve the correct passphrase during a login process, a user usually will have to know the service to which he/she is currently authenticating. The service usually requests a a service-local identifier for the account before a passphrase is requested. Thus, to identify the right login and passphrase a reverse mapping from service name is required.

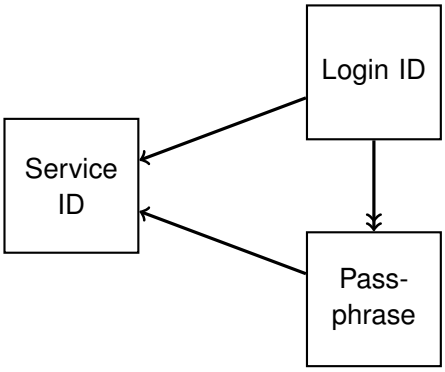


Figure 4.5: Simple OLOG of password credential data.

In Figure 4.5 the different maps between service, login and passphrase are shown. A user may have different login/passphrase pairs for a single service, but for every login/passphrase there is exactly one service for which they are intended. (Although syntactically the login name used in different services could

be equal.) It should further be possible to map from each login to the related passphrase. This map is drawn as a surjective map, although there is a one-on-one relation between login and passphrase, to emphasize that it should not be possible to derive a login from a passphrase.

From this data-structure we can derive a few requirements for storing of passphrases. We require a reverse map that allows to retrieve a set of login/passphrase pairs for a known service. (As it is possible to have multiple accounts.) There further has to be a way to retrieve the passphrase for each login.

Necessarily, only the service ID has to be stored in a searchable (e. g., associative) way. It is not necessary to store the login identifiers searchable, i. e., readable, they could be encrypted. Logins identifiers should be handled as a shared secret between service and user. Passphrases are private secrets of users and should never be stored or communicated in the clear.

Passphrase Generation

The main attributes of a password are that it is hard to be guessed by an attacker but easy to remember by an authorized user.

Random password generation is actually not to complicated. You need a source of randomness and translate it into passwords by sequentially selecting letters from an alphabet.

- Manual Methods and Problems
  - Hit the keyboard → typing behaviour
  - Think of letters → Unconcious patterns
  - Flick through a book → language bias
- Generate 23-letter-password:
  - Random password `pwgen -sy 23 1`
  - Password manager `pass generate <pass name> 23`

Exercise 6

Generate “random” words by hitting at your keyboard. Generate 100 words of at least 8 letters length.

1. Can you spot any patterns?
2. Make a frequency analysis of letters and digraphs and plot the results in a frequency graph, sorted by frequency. Which stochastic model describes the graph best?

Figure 4.6: 30-sided dice with german umlauts and double letter 'E'

3. Calculate the entropy estimate of your randomness source based on the digraph frequency. What would be the best possible entropy you could gain?

Of course, random letters might be difficult to guess, but they are also hard to remember. Our brains simply are not normally wired that way, although it is possible to train them[3].

Diceware Passphrases

Another way for random passphrases, that is a little more suited to our brains, is to use words in a dictionary as "letters" and create longer passphrases by randomly choosing a sequence of words as passphrase. As with normal passwords, the resistance against guessing-attacks is based on the number of possible combinations. It is crucial to note, that secrecy of the word-list is not part of the security scheme.

[3] Few of you might remember that people where able to remember a large list of telephone numbers, before telephones learned to include telephone-books. Make it a habit to dial at least the two or three most important numbers for you by hand, it comes in handy if you drain the battery of your phone.

Diceware Passphrases use a number of throws of a 6-sided dice to generate randomness and select words from a fixed wordlist. The original wordlist by Arnold G. Reinhold contain $6^5 = 15625$ words. A main security feature is, that the whole process is manual, i. e., no — potentially compromised — computer system is involved[4]. The idea might seem funny, but it produces secure and somewhat memorable passphrases and thus has become part of the security toolkit[5]

In order to generate a passphrase, you need to throw the dice five times per word, find the words inside the wordlist and note them down. A recommendation is to use passphrases of at least six words.

A human can memorize these words easier than random letters, because the human brain understands the meaning of the words and can much easier store a connecting story for them. This actually is otherwise a technique of people who aim to memorize long sequences or numbers in memoization contests. Diceware Passwords are a good choice if you plan memorize the generated passphrases, want to store them on paper and type them manually. If you want to store and handle them digitally it has no advantages over sequences of random letters.

Manual Password Generation Algorithm

If it is hard to remember a password for every account you have somewhere, especially for services you might be using rarely. If you don't want to write your passwords down, maybe you could memorize a single algorithm with which to construct — seemingly random — passwords from the context of the service.

- Memorizable
- Manually computable (ideally in the head)
- Individual (use context)
- Hard to guess (include master secret)
- Hard to guess knowing pwds from other services

There are some problems with this approach. Firstly, you must be able to calculate your password easily in your head. Thus the algorithm has to be somewhat simple. Usually large parts of the algorithm will have to be secret, so you will

[4] But obviously many people, including me, could not keep themselves from implementing dice-word generators. Although, using a computer, it would make sense to get rid of the restrictions imposed by the dice and use arbitrarily long wordlist.

[5] Much funnier is, that Mira Modi, at that time eleven years old, became — arguably — more famous than the inventor for selling diceware passwords. She promises to generate all passwords by hand, using her "Top Secret Business Passwords"-list put them on a linnen paper in a linen envelope — and to forget them afterwards. US. Postal Service only, sorry.

have to design your own algorithm[6]. Therefore your passwords for different services necessarily have some common structure, even if you yourself are not able to see it. After all, everybody can construct a crypto-scheme that he himself is not able to break. As a result, your other passwords become less secure if a password from any of your accounts is disclosed.

Secondly, accounts often require more than a password, for example a login name, that you cannot always choose freely. For example the login name identifies an individual account and thus has to be unique within every given service. Some services require secret answers alternative "security questions"[7]. Thus you most likely need to store some secret login-related data somewhere.

But, if you require some support in memorizing certain passwords, especially passwords that you need to enter physically into a keyboard, then password generation algorithms might be a good way to memorize them.

The general approach is to apply a secret sequence of syntactic operations to mix up secret phrases with the an identifier of the service. The result should contain letters from all allowed classes of letters, e. g., alpha-numerical plus punctuation characters. Your algorithm should not produce results that are contained in common wordlists. The identifier of the service must not be recognisable in the result, otherwise an adversary could easily guess your password for other services.

The method itself is not fully recommended, but a few hints on how to build an algorithm:

- Take first/last/middle letters of domain name
- Transpose these letters (e. g., ROT-1)
- Append Secret Word (master-password)
- Append special character
- Permute some letters (to scramble secret)
  - e. g., n = length of domain name
  - count to nth letter
  - count n letters further (maybe wrap word)
  - exchange both letters
  - repeat

[6] Using secret algorithm is directly against the second of Kerckhoff's principles for cryptographic algorithms. That rarely is a good thing.

[7] I hereby must note, that these alternative security questions are nothing more than alternative passcodes to your account and do not increase security, but often pose themselves grave security risk.

- swap some letters for numbers
- capitalize letters at some positions

(Motivated by Password Algorthms by Sumit Khanna, 2017-10-24)

Password Managers

With the requirement to choose random passphrases and never reuse a single passphrase, remembering becomes a problem. Luckily humanity has invented writing a long time ago and thus one of the most basic ways to remember passphrases is to "just write them down". The simplest way is to use pen and paper and notch everything down in a notebook[8].

Obviously the notebok becomes a critical asset whose contents now need to be handled with an increased secrecy, accessible only to yourself. But one big advantage of paper-storage is that any access requires physical co-presence, i. e., it is (generally) save from remote attacks. A disadvantage is, that is lacking the comforts of digital storages that we have become used to — and that actually are the main reason you actually need a passphrase store.

Thus, to no surprise, the idea is to use digital notepads to notch down all your passwords. And, while at it, why not try to utilize the whole set of available functionality that makes digital data handling — as compared to plain old paper — comfortable? Namely searchability, single-sign-on, browser/tool-integration and synchronisation between multiple devices.

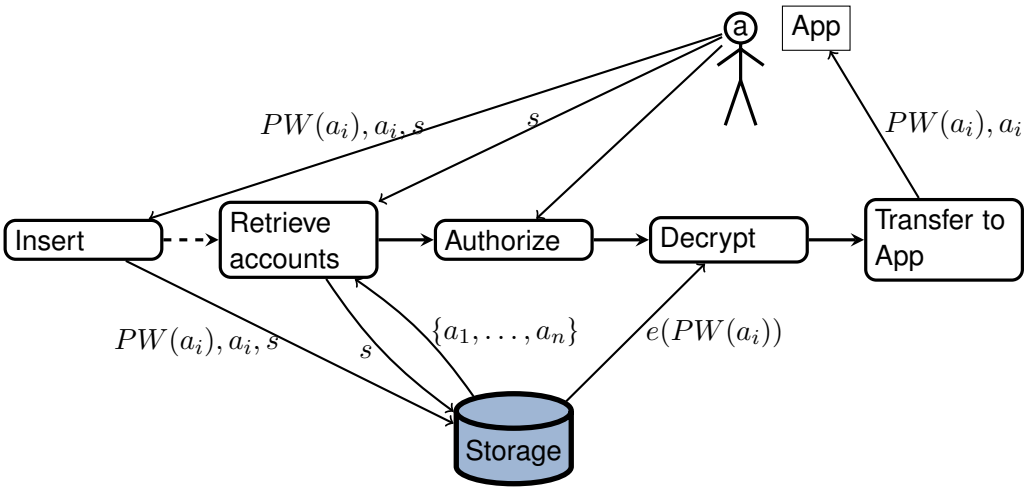


Figure 4.7: Using a password manager

[8] keeping a journal-like scetchbook close to you never has been a bad idea

Name	**OS**	**Pl**	**Enc**	**Sync**
pass	Yes	most known plattforms via (G)UI clients	GnuPG	git
KeePass	Yes	via contributed/inofficial ports	Yes	local file
passbolt	(Yes) Community Version	Linux/CentOS	central cloud store	cloud
vaultwarden		Web		server

Table 4.1: Selection of Password Managers (OS=OpenSource, Pl=Plattform, Sync=Synchronization)

But, security first! Using digital password managers means your most precious passphrases are now available on one or more computers. We are not going to dig into “building a secure password manager”, but you should be aware of some crucial distinctions between the ways your passwords are stored and handled.

- Encrypted storage
- Local vs. remote storage (i. e., cloud)
- Transfer from storage to login form
  - Temporary storage (e. g., clipboard) whiped after use?
  - Direct access through applications (e. g., browser)?
- Added security services
  - Random passphrase generator
  - Passphrase renewal reminder
- Synchronization between devices

With your pen-and-paper notebook the same question arises as with your digital password manager: where to store the data securely. Given that your storage uses secure encryption, the first line of defence is that the storage is not accessible from the outside so that an adversary cannot start brute-forcing your master password right away.

There are a few different *password managers* available already.

HowTo Use pass

As an example for the usage of a password manager, we will take a look at pass, the standard unix password manager. The program makes extensive use of existing tools, uses GnuPG for encryption and, optionally, git for version control and synchronisation. It is itself a comparatively short program of less than 800 lines of bash-Skript. Around it a whole community of additional user-interfaces, graphical and command-line, has been developed. It can be integrated directly within many applications or used on a wide range of window managers and operating systems.

In the following I will give a very short introduction into the basic process of setting up a password-storage, inserting and retrieving passwords. For everything else please consult the excellent documentation on the man-page.

For a start you need to install GnuPG as pass relies on it for encrytion. (If you use a Linux Distribution this, most likely, will happen automatically during installation.) Git is optional, but if you want to remember old passwords after changing them, you'll need it. Also, if you maybe want to synchronize your store, a remote git-repository is a simple and proven way to do so.

1. Generate a new PGP-key with gnupg
2. Create a directory for the store (e. g., `/home/me/.passwordstore`)
3. Init the store: `pass init <storedir> <GPG-key-id>`
4. Init git (optional): `pass git init`

After initialisation you can start to add or search and retrieve passwords or multi-line text. The password store essentially is a bunch of encrypted files within a directory tree, that you address as path starting at the passwordstore-directory. The pathname itself is readily searchable by everyone with read-access to the directory, the contents of the files — obviously — are not.

It is wise to take a few minutes to think about how you intend to access your passwords. Basically there are two ways: using service- and login-name as part of the search path, or only using the service-name in the file path and encrypting login-name and passphrase in the file. Take a look at the website for an example.

If you store not only one single password-line in an item, than you probably want to create some syntax to distinguish passphrase and login-name. One way could be to have the login-name in the first line and the passphrase in the second line of a multiline file in the store.

- Blind entry
    1. `pass insert`

2. Enter passphrase
3. Confirm by entering again

- Visible entry
  1. `pass insert -e`
  2. Enter passphrase
- Multiline: use parameter `-m`

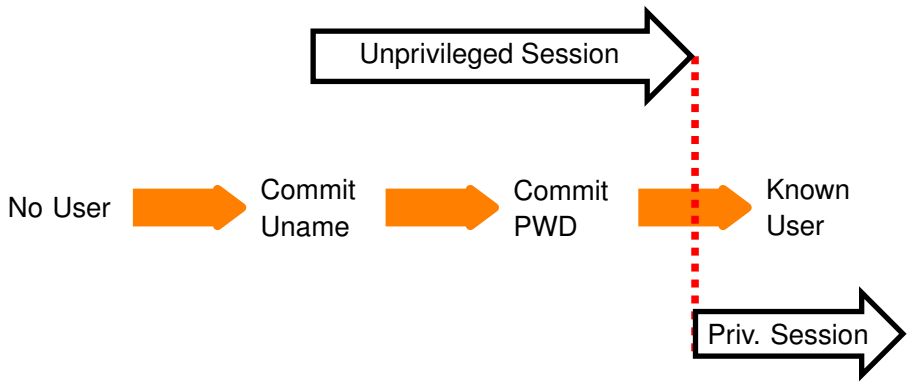
To use the contents of your password you have to know where in your password store the needed password is to be found. The best thing is if the hierarchy of your store is so well organised that you can directly enter the path of the needed file. Otherwise, use the search function.

The password is retrieved by the show-command, which is the default if you give no command.

- Searching in path-name item `pass search <something>`
- Retrieving `pass <path>`
- Retrieving into clipboard:
  1. `pass -c <path>`
  2. Passphrase of the GnuPG-Key asked interactively
  3. Clipboard will be wiped after 45 seconds.

4.1.6 Server-side Password Handling

Passwords are never directly stored in cleartext in a database (or anywhere else). In order to verify passphrases, cryptographic hashes are used. Only the hash of a passphrase is stored with the verifier. To verify that a passphrase entered by an entity the hash of the entered phrase is tested against the stored hash.



That way the possibility that a threat agent (including the verifier) can make use of a password database is limited.

- Brute-force: Test every possible password/message, until you find one that fits. On average you will have to try half of the possible passwords. How many are this for a 512-bit Hash value?
- Dictionary Attacks: Trying lists of common-passwords or using common password-generating methods to "brute-force" the most likely passwords first. Can you find a list of common passwords?
- Rainbow Tables: Pre-compute all possible passwords/messages, i. e., create a big table of (at least one) message for each hash value. Storage is cheap and in that way finding a matching password/message to a given hash is a fast and simple lookup in a huge table. How much storage is required for a rainbow-table?

Salting Passwords

In order to prevent Rainbow-Table attacks, an additional secret can be added to a password before the hash-function is applied. This secret is called a "salt" and must be known only to the authentication function, i. e., the host that verifies passwords.

In that way, the password known by users, and entered into the login process, is not the message used to generate the hash value. Thus, an attacker gaining knowledge of a list of password hashes cannot simply lookup the corresponding password in a rainbow table, but has to know the used salt.

The salting process, as is schematically shown in Figure 4.8, can add the salt in different ways to the password. Commonly the salt is simply appended to the passphrase-string. Binary XOR would be another, more secure variant, as it basically implements a One-Time-Pad. Another version would be the use of HMAC.

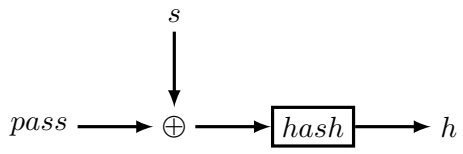


Figure 4.8: Salting a password before it is hashed.

A plethora of even more stupid password-equivalent ways to provide authentication are:

- Strange Password Rule Enforcement
  - *Change every 5 days with at least two changed letters.*

- “Security” Questions
  - *Please post 5 personel questions and answers.*
- Pre-Set “Security” Questions
  - *Your mothers maiden name?*
- Security Questions that are not verified[9]

There are some guidelines, found, for example, in the “Grundschutz Handbuch” of the “Bundesamt für Sicherheit in der Informationstechnik” Section 2.11[10].

Administrators should

- Filter trivial passwords (e. g., “123456789”)
- User-changeable at any time
- First-use/enrollment: use One-Time Passwords
- Login-failures: boolean error message, increase wait time
- No unencrypted pwd transmission
- No pwd display on screen
- Store pwds “encrypted” (one-way fkt)
- Prevent pwd-reuse of old pwds

This is essentially in conflict with the principle that security should not rely on the strength of the attacker.

4.2 Token Authentication

Authentication Tokens are physical objects that can be carried and proof authentication by ownership. If implemented as active devices, e. g., SmartCards, they can provide *two-factor authentication* by utilising knowledge of a secret, e. g., a PIN.

Token = Object/Possession to autheticate

- embossed card
- magnetic stripe card
- memory card

[9] `http://it.slashdot.org/story/12/08/09/1410231/secret-security-questions-are-a-joke`, 2013-07-03

[10] `https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02011.html`, 2013-12-06

- smartcard
  - RSA-ID Token
  - nPA
  - Chip-TAN

Smart Cards

ISO/IEC 7810 and ISO/IEC 7816

define size and protocol for smart cards.

One common type of smart cards are the EMV-Cards (abbreviation for Europay, MasterCard and Visa) widely used for payment.

PGP-Cryptosmartcards

MiFare card

Common Attacks: Yes-Card-Attack (MitM Card replies yes to any PIN),

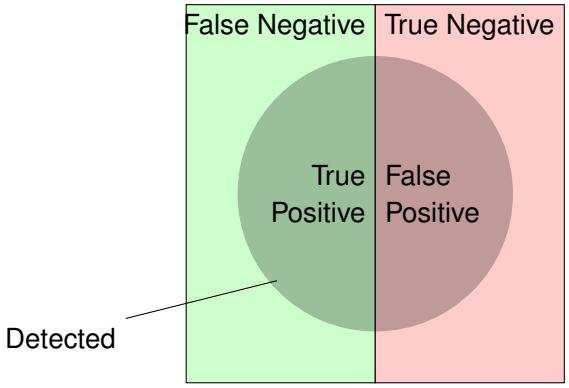
`https://www.cl.cam.ac.uk/research/security/banking`

4.3 Biometric Authentication

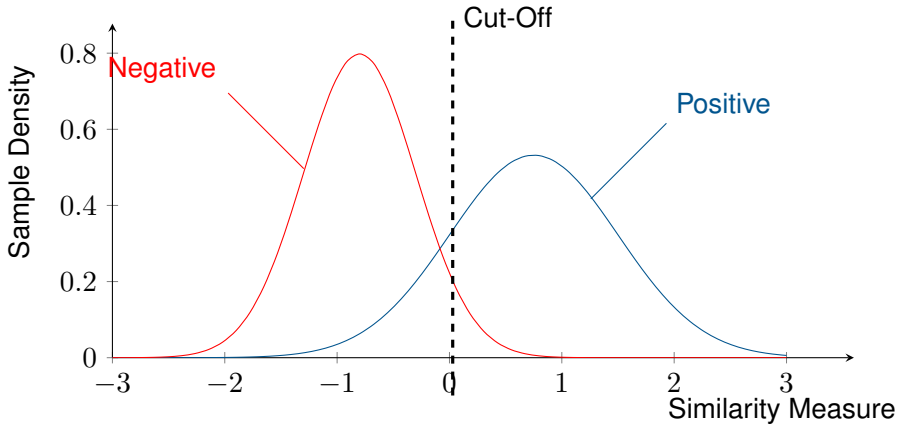
- Measuring human features
  - Physical
    - Fingerprint
    - Iris Scan
    - Brainwaves...
  - Behavioral
    - Typing Rhythm/Speed
    - Signature/Handwriting
- Impact of Biometry

Generally biometric authentication is a statistical binary classification test, i. e., a biometric scan leads to one of two possible results: rejection or acceptance. But biometry is inherently error-prone.

A biometric test can essentially fail in two ways: falsely rejecting a valid authentication and falsely accepting an invalid authentication. The termini for these errors are *false negative* and *false positive*, or Type II and Type I error. Usually a test method always has a probability for both types errors to a certain degree which can be adjusted by a threshold value. There are different ways to quantify the errors made. The most generic are Sensitivity and Specifity.



Any result of a binary classification consists of two distinct sets of samples which a classification mechanism should distinguish. A classification algorithms detects a set of samples, i. e., classifies them as “positive”. The dual is the set of undetected samples, which is thus classified as “negative”. Within the set of detected samples we denote the correctly detected samples as “true positive” (TP) and the set of wrongly detected samples as “false positive” (FP). Similarily the set of undetected samples is comprised of “true negative” (TN) and “false negative” (FN).



Sensitivity (true positive rate) $TPR = \frac{TP}{P}$

Specifity (true negative rate) $TNR = \frac{TN}{N}$

- Misclassification is unavoidable
- Persistent personal attributes
  - Cannot be changed (easily)
- Cloned/Faked Attributes, e. g., 2008: Schäubles Fingerabdruck
- Dangerous Data-Leaks

- Unauthorized use: e. g., 2020: Taliban get hold of biometric database of local forces

Biometrical attributes differ in main aspects from knowledge and possesion as authentication methods. Namely they cannot easily be denied or changed by the individual authenticated by them. Changing biometrical features always means substantial change to body or behaviour which usually requires surgery to alter attributes of the body.

A consequence is, that biometrical attributes are unique to the individual and are similar to using a single password for everything. Which can be problematic as security research has constantly found ways to cheat sensors or algorithms for biometric attributes.

insert references from starbugs work on fingerprints, iris and face recognition

Furthermore the biometrical information stored in databases can be used for other purposes, e. g., yield information about health and genealogy, or can be used for identification without consent. Recent history sadly provides us with examples with potentially catastrophic consequences for the individual.

During the retreat of western forces from Afghanistan, Taliban Have Seized U.S. Military Biometrics Devices (The Intercept, 2021-08-18). The stored information on these devices possibly can be extracted and used to identify people that have worked with the western forces, which, if history repeats, will put them and their relatives and acquaintances in great mortal danger of retributions. Thus a tool for biometric authentication has been turned into a tool for identification. Different from other authentication methods, the chances of individuals avoiding identification from iris-scan or finger-print-scans involve dire mutilation of their bodies.

4.4 Challenge-Response

The term *Challenge-Response* describes a method used for authentication where the verifier sends a challenge to the entity that wants to authenticate itself. The entity then responds with an answer that proves that it has knowledge about some secret. The secret can be symmetric or asymmetric.

Challenge-response techniques can be found in symmetric and asymmetric flavours. The concept is very simple. The verifier generates a (random) challenge for the prover. The prover then calculates something based on the challenge that shows his knowledge of a secret. This, compared to static passwords, has the additional advantage of providing some proof of freshness.

Mutual Challenge-Response

1. Server sends unique challenge sc to client
2. Client generates unique challenge value cc

3. Client computes $cr = hash(cc + sc + secret)$
4. Client sends cr and cc to the server
5. Server calculates the expected value of cr and compares
6. Server computes $sr = hash(sc + cc + secret)$
7. Server sends sr
8. Client calculates expected value of sr and compares

- Challenge: Not Predictable, Fresh Nonce
  - e. g., random + datetime
- Response: Application of Secret
  - e. g., signature, HMAC, encryption

CR with symmetric/assymmetric crypto

4.5 Authentication of Certificates

The most prominent application is certificates used for authentication, i. e., certification of the relation between an identification and a public-key pair. Thus, a certificate can be used for verification that a webserver is correctly serving under the identity of `www.example.org`, or that the sender is indeed *jon.doe@example.com*. The two remaining questions are: who is the issuer of the certificate and why can the issuer be trusted to know that the identified holder of the certificate is authentic.

Digital certificates are omnipresent. They are needed for secure communication or to prove access rights and qualifications. But, what exactly is a “digital certificate” in IT-Security?

A certificate

- relates list of **attributes**
- authenticity of **provable authority**
- allows proof of **ownership**

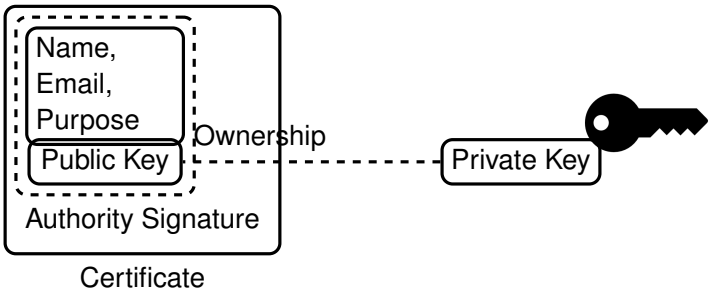
Example: Passport

Attributes
- Name,
- Birthday,
- Place of Birth,
- Unique ID,. . .

Provable Authority
- Security Features,

- official seal,
- well-known design

Ownership
- Biometric data,
- physical ownership



Certificates are secured by digital signature of an issuer. The first question regards the authenticity of the issuer of the certificate. Of course an issuer can be authenticated by providing a certificate, but this is essentially the same question for which the original certificate should have solved. Essentially this is usually solved by another a chain of certificates that leads to a trusted source.

The second question is obviously not only solved in the digital domain, as trust is foremost a human concept. [**Josang2003**]

4.5.1 Authentication Metrics

[**Reiter1999**] provides principles how authenticity should be measured.

4.5.2 Public Key Infrastructures

A *Public Key Infrastructure* (PKI) denotes a hierarchical structure to issue, distribute and verify certificates.

Cross Certification means the mutual authorisation of certification authorities that expresses trust in the other CA and enables usage of signatures from one PKI tree within another. In practise this is implemented by providing a second certificate to a self-signed root-certificate.

Take for example two CA A and B who have established mutual trust. That means that CA A authorizes all certificates signed by CA B and vice versa. This would establish an alternative trust-paths for the certificates signed by any of the two CA.

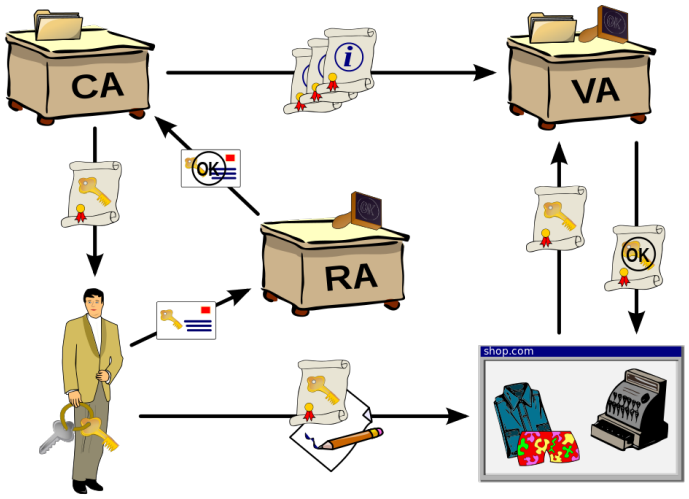


Figure 4.9: PKI Participants

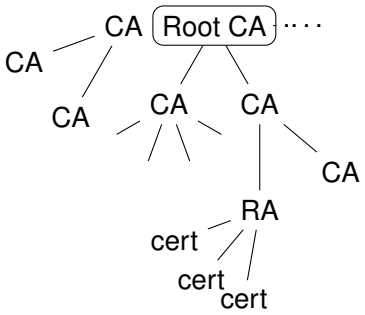
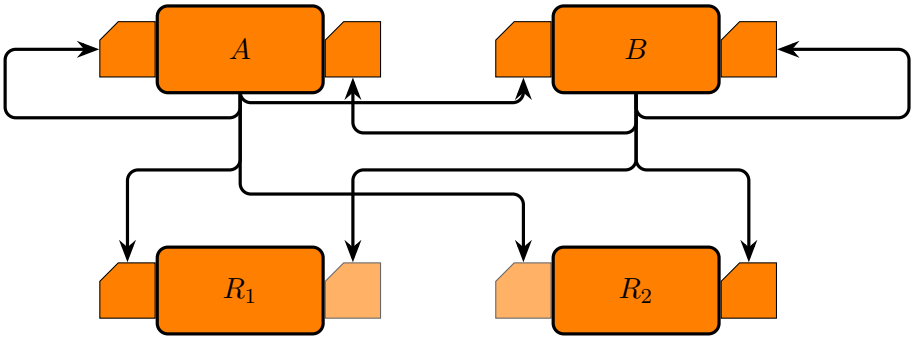


Figure 4.10: Tree of Certificates



X.509 Certificates

In this section we introduce the structure of X.509 certificates version 3. X.509 is an ITU-T[11] standard that defines a Public Key Infrastructure (PKI) and a Privilege Management Infrastructure (PMI). The PKI is expressed by certificates that bind identities to public keys by way of signatures of certification authorities (CA).

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (opt)
  - Subject Unique Identifier (opt)
  - Extensions (opt)
- Certificate Signature Algorithm

[11] International Telecommunication Union — Telecommunication Standardization Sector

- Certificate Signature

Certificate Revocation List

Certificate Revocation Lists (CRL) provide revocation of certificates in the X.509 PKI scheme. A revocation list is published by a CA and contains identifiers of certificates that became invalid before their validity time expired. CRL are standardised in RFC 5280.

There are various circumstances under which a certificate has to be revoked. Among others there are: Revocation vs. Hold

- Key Compromise The private key of the holder of a certificate has been disclosed to an attacker. Even the suspicion of a compromised key might be sufficient reason to order a re-certification.
- CA Compromise the issuing certification authority has been compromised and the currently signed certificates may have been faked. This obviously is a very big issue and there might be a lot of revocation certificates be circulating at the same time. Probably all certificates signed by this authority are affected.
- Privilege Withdrawn means that a privilege that has been granted by a certificate, for example the privilege to sign certificates, has been withdrawn from the holder — possibly a CA.

Critique of PKI

- Impersonal Trust Relation
- Difficult to Control Trustworthiness of CA
- Certificate Revocation is Slow and Blacklist-based
- Complicated Protocol for CRL/Cert. Distribution
- X.509 is complex
- X.500 Identifiers complex and not widely used

4.5.3 Web of Trust

The *Web of Trust* (WoT) is an alternative concept to PKI that is not based on institutionalised trust relations, but only on direct trust.

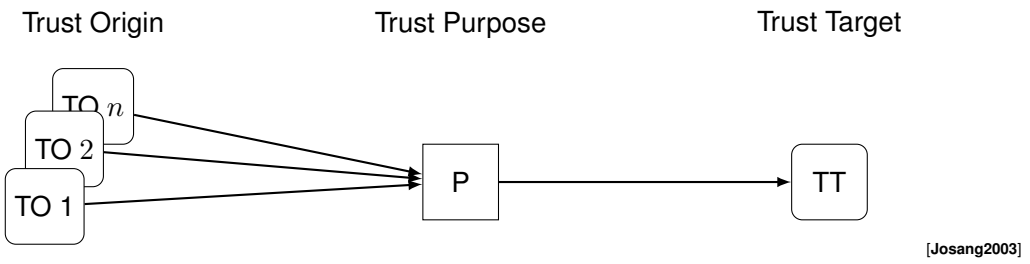
- Phil Zimmerman
- Traditional Format RFC 1991 `https://www.ietf.org/rfc/rfc1991.txt`

- MIME/PGP RFC 2047 RFC 3156
- Authentication of Credentials
- Without direct (secure) contact
- Everybody may sign a public key and thus create a certificate
- Trust only trusted persons' certificates

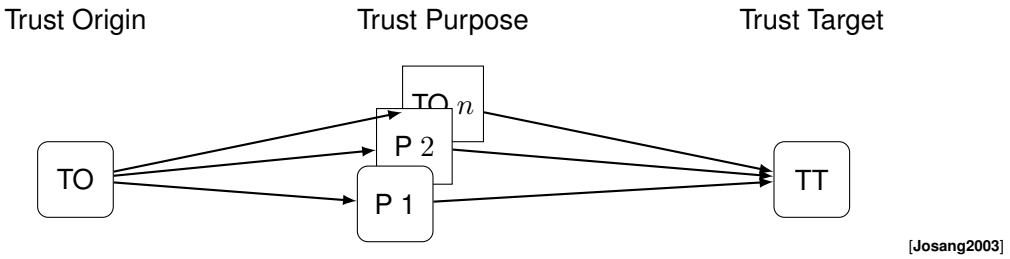
Trust

> Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.[**mcknight1996:meaningsoftrust**]

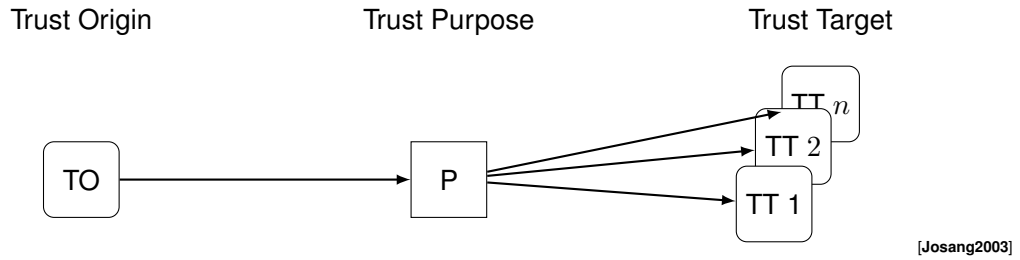
Remark: Negativ consequences are possible, because the outcome is not under the party's control.



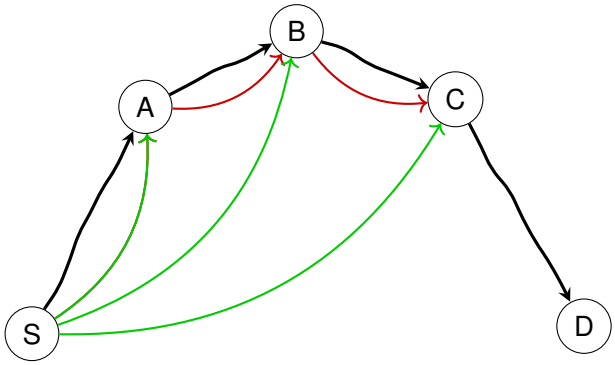
[**Josang2003**]



[**Josang2003**]



[**Josang2003**]



Transitive Trust Direct Trust

A key is valid if:

1. Sufficient number of trusted signatures
2. Chain of certificates

untrusted Signatures by this key are ignored

marginal At least n signatures of marginally trusted keys are needed to make a key valid. A common value for n is two.

complete One or more signatures of completely trusted keys are sufficient to trust a key. This value also can be configured locally

ultimate To the owner of a secret key, the connected public keys are ultimately trusted.

PGP Signatures

A list of signatures that signs the key `65D0FD58` in Figure 4.11 contains various attributes of signatures. In general, a signature expresses some degree of certainty that some person, identified by email and name is holding the matching private key to the certified public key.

numbers 1-3 for certificate check level

0 means you make no particular claim as to how carefully you verified the key.

1 means you believe the key is owned by the person who claims to own it but you could not, or did not verify the key at all. This is useful for a ”persona” verification, where you sign the key of a pseudonymous user.

2 means you did casual verification of the key. For example, this could mean that you verified the key fingerprint and checked the user ID

```
pub   rsa2048 2020-06-05 [SC] [expires: 2022-06-05]
      583263EAF239F5802E3786F1D48973AA9B0F4D73
uid   [ unknown] Lars Fischer <lars.fischer@hs-bremerhaven.de>
sig 3 D48973AA9B0F4D73 2020-07-17  Lars Fischer <lars.fischer@hs-bremerhaven.de>
sig   ABD2FB41822CF4D1 2020-07-27  Lars Fischer <lars.fischer@hs-bremerhaven.de>
uid   [ unknown] Lars Fischer <lafischer@hs-bremerhaven.de>
sub   rsa2048 2020-06-05 [E] [expires: 2022-06-05]
sig     D48973AA9B0F4D73 2020-06-05  Lars Fischer <lars.fischer@hs-bremerhaven.de>
```

Figure 4.11: Example PGP Signature List

on the key against a photo ID.

3 means you did extensive verification of the key. For example, this could mean that you verified the key fingerprint with the owner of the key in person, and that you checked, by means of a hard to forge document with a photo ID (such as a passport) that the name of the key owner matches the name in the user ID on the key, and finally that you verified (by exchange of email) that the email address on the key belongs to the key owner.

"L" for a local or non-exportable signature,

"R" for a nonRevocable signature,

"P" for a signature that contains a policy URL,

"N" for a signature that contains a notation,

"X" for an eXpired signature,

numbers 1-9 or "T" for 10 and above to indicate trust signature levels that combine the notions of certification and trust in a signature. This is probably only useful in closed groups.

4.5.4 Simple Public Key Infrastructure

Due to restrictions in space-time we will not introduce the *Simple Public-Key Infrastucture* (SPKI) here. But you are very free to research it for yourself. You will find sources with the SPKI working group[12] of the IETF. Or you directly take a glimpse into the standards they developed: RFC 2692[13] and RFC 2693[14]. We apologise for this inconvenience.

[12] http://www.ietf.org/html.charters/spki-charter.html
[13] https://tools.ietf.org/html/rfc2692
[14] https://tools.ietf.org/html/rfc2693

8

Secure Communication

Contents

Lernziele:

- Konzept: Hybride Protokolle
- Perfect Forward Secrecy
- Attacks
  - MitM
  - Replay
  - Session Hijacking/Pinning
- Prudent Engineering Practice
- TLS
- Key Distribution (PKI)

Cryptography alone is no guarantee for security, it must be deployed in a sensible way to achieve the intended security objectives. In this chapter we present example communication protocols that are selected for their importance in existing networks or for their teaching value. Without yet concerning ourselves with implementation errors, the communication protocols that utilise cryptography produced series of vulnerabilities as well.

Include RFC 2223 Instructions to RFC Authors

Split into Lecture on "Protocol Design" and "Protocols"

> Principle 1: Every message should say what it means: the interpretation of the message should *depend only on its content*.
>
> It should be possible to write down a straightforward English sentence describing the content though if there is a suitable formalism available that is good too.
>
> [**abadi96prudent**]

□

Participants A, B, C, I
Messaging $M \quad A \longrightarrow B : M$
Keys: Symmetric, Public, Private $K_{a,b}, K_a, K_a^{-1}$
Encrypted Message $\{M\}_K$
Signed Message $\{M\}_{K^{-1}}$

Figure 8.1: Notation for Cryptographic Protocols

8.1 Hybrid Encryption Protocols

Asymmetric encryption tends to require costly computation, as compared to symmetric encryption algorithms.

Problem: Asymmetric Algorithms are costly Solution:

- Use symmetric encription for data
- Session Key

The solution is to use a *session key*, i. e., a key valid and used only for a single conversation session, or even only a part of that conversation. Session keys are re-established often. Establishment can be done by a single principal or via shared key-establishment protocols, e. g., Diffie-Hellman (see Section **??**).

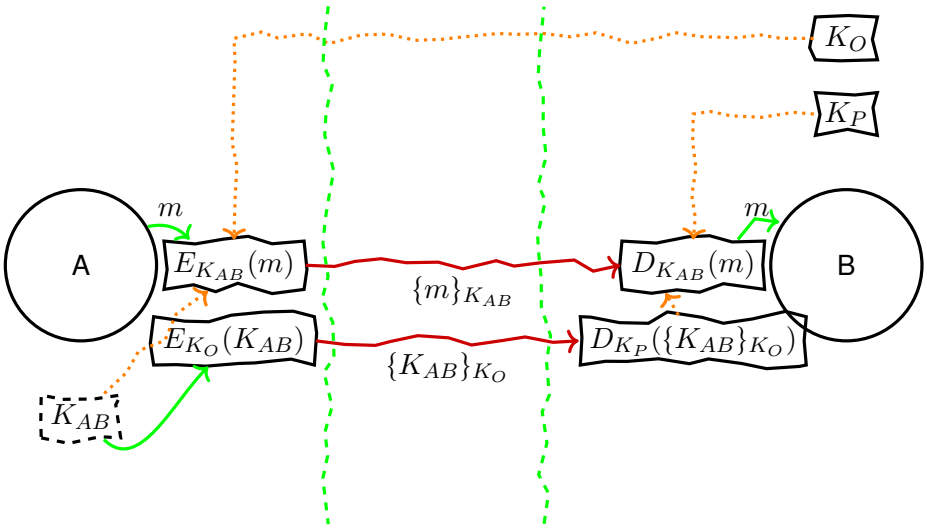


Figure 8.2: Hybrid encryption scheme, using symmetric session key K_{AB} for encryption of data m. Key is encrypted by asymmetric algorithm.

8.2 Security Attributes

8.2.1 Perfect Forward Secrecy (PFS)

Problem: Compromise of master key reveals all communication Solution:

- Temporary valid keys
- Exchange/Generate at start of session
- Regular re-establishment, commonly the Diffie-Hellman protocol is used for this.
- (Authenticated!)

Perfect Forward Secrecy (PFS) means compromise of a master key will only compromise future (and at most current) data exchanges. Compromise of session keys will only compromise current exchange.

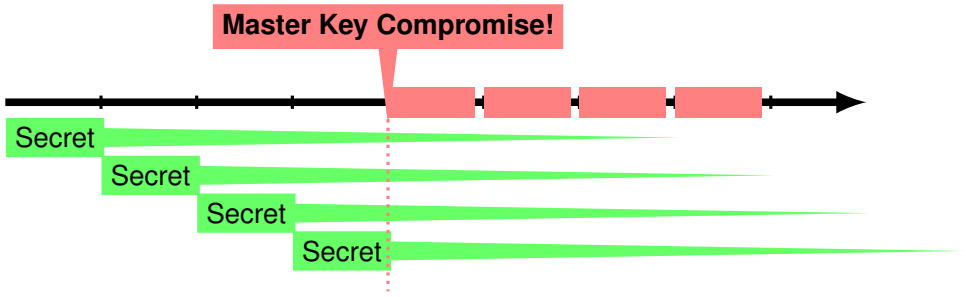


Figure 8.3: Compromised communication without Perfect Forward Secrecy (PFS)

To achieve PFS it is required that every session is protected by an independent, random secret session key which is not reused. This secret session key has to be exchanged between all authorized principals. It is crucial that all principals verify the authenticity of the session keys received to prevent Person-in-the-Middle (PitM) attacks.

8.3 Transport Layer Security (TLS)

At this point we finally went through all the necessary prerequisites to take a look onto real world examples for secure communication. This section is based on Chapter **??** and the previous sections of this chapter here.

Transport Layer Security (TLS) is a protocol that provide secure communication, i. e., secret and authenticated, directly on top of some reliable transport protocol, e. g., implemented by TCP. In the layer model it is situated between TCP/IP (i. e., the 3rd and 4th layer) and the application layer. (See Figure 8.4)

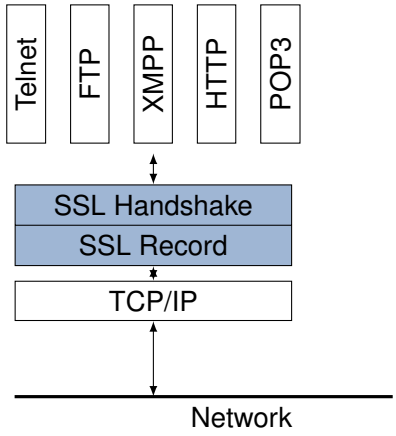


Figure 8.4: TLS Layer Model

The TLS protocol is currently standardised in version 1.2 by RFC 5246. The protocol has a history of three major revisions as Secure Socket Layer (SSL) that has been introduced by the almost forgotten now browser company Netscape.

TLS is comprised of two sub-protocols, the TLS Handshake and the TLS Record Protocol. Basically the former handles the establishment of a secure communication session and the latter handles the actual data transfer. Both sub-protocols are comprised of multiple separate messages.

TLS already is defined for a number of different cryptographic algorithms, which can be replaced by newer algorithms in the future.

- Anonymous/Uni-/Bi-directional Authentication
- HashAlgorithms: none(0), md5(1), sha1(2), sha224(3), sha256(4), sha384(5), sha512(6), (255)
- Signature Algorithms: anonymous(0), rsa(1), dsa(2), ecdsa(3), (255)

8.3.1 TLS Handshake Protocols

- Negotiate Algorithms, Random Values, Session Resumption
- Exchange Cryptographic Parameters
- Exchange Certificates
- Generate Master Secret
- Verify Integrity of Handshake Protocol Run

ClientHello First Message send to initiate a connection. Contains: Random structure: GMT and 28 random bytes, Session ID (Empty or old ID), list

of supported cipher suites, compression methods, and extensions.

ServerHello Response to ClientHello. Contains essentially the same fields.

HelloRequest

ServerHelloDone Indicates the end of the server-side of the handshake, and that the server will now wait for the client's response.

Server-/ClientCertificate Contains a chain of certificates for authentication. These messages could already contain Diffie-Hellman parameters for the generation of the master secret.

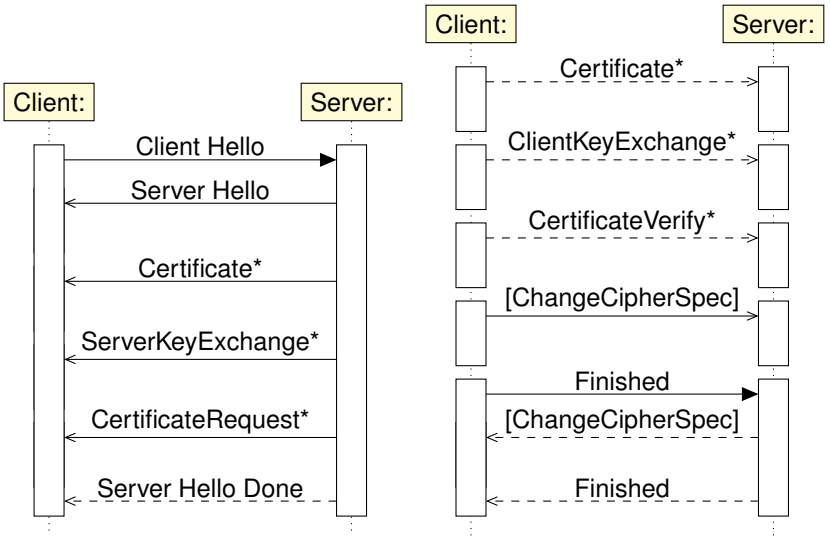
CertificateRequest Message from the server to request TLS authentication from the client, if the server is non-anonymous. Contains lists of acceptable certificate types, signature algorithms and certificate authorities, the latter could be empty to indicate "any".

CertificateVerify Signature by the client over all handshake messages exchanged so far.

Server-/ClientKeyExchange Exchange of Diffie-Hellman Parameters (if not already included in Certificates).

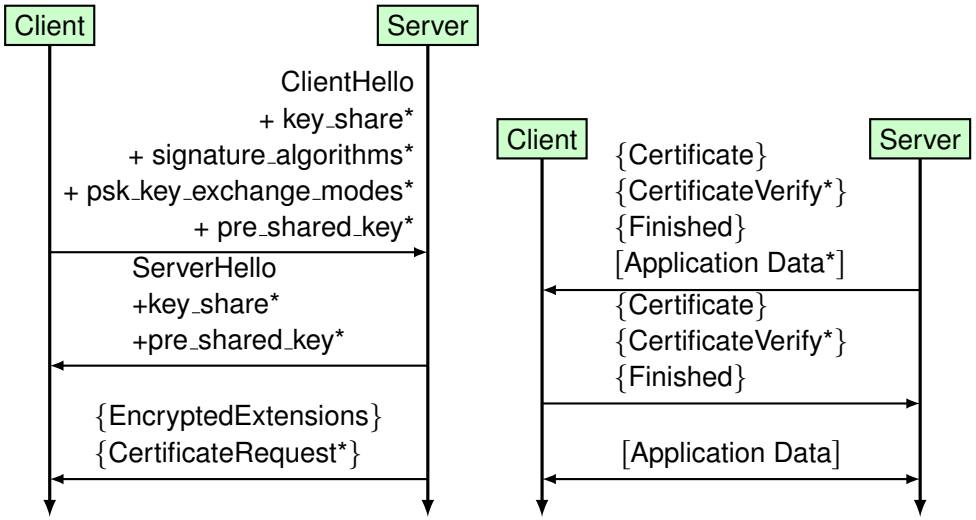
ChangeCipherSpec Last message before switching to the selected encryption.

Finish These are the first messages exchanged in the negotiated secure connection. This message is essential for the whole negotiation as it includes a hash value over all handover messages and the master key.



The most noteworthy change in version 1.3 of the TLS Handshake Protocol

is — arguably — that the handshake messages after the ServerHello are encrypted.



More informally the whole process can be described in six steps:

1. Exchange hello messages to agree on algorithms, exchange random values, and check for session resumption.
2. Exchange the necessary cryptographic parameters to allow the client and server to agree on a premaster secret.
3. Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
4. Generate a master secret from the premaster secret and exchanged random values.
5. Provide security parameters to the record layer.
6. Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

[**rfc5246**]

TLS Certificate Request

- CertificateRequest
- immediately after ServerKeyExchange
- WebID: empty `certificate_authorities`

```
struct {
    ClientCertificateType certificate_types<1..2^8-1>;
    SignatureAndHashAlgorithm
      supported_signature_algorithms<2^16-1>;
    DistinguishedName certificate_authorities<0..2^16-1>;
} CertificateRequest;
```

TLS Client Verify

Client verification is used to prove that the client is in the possession of a private key associated with an identifier. Normally this identifier is bound to the individual person assumed to sit in front of and controlling the client. Thus the prerequisite to user authentication using this is, that users do not share browsers (or at least not their key rings).

Client Verification is prominently used in WebID described in Section **??**.

- immediately after Client Certificate
- must have signing capabilities

```
struct {
  digitally-signed struct {
     opaque handshake_messages[handshake_messages_length];
  }
} CertificateVerify;
```

TLS Key Generation

A Pseudo-Random Function (PRF) as utilised in TLS must be able to generate a stream of pseudo-random numbers. The PRF used for all standards up to TLS 1.2 is defined as follows.

The PRF is based on a recursively defined application of a SHA-256 based HMAC:

$$\begin{aligned} P_{hash}(secret, seed) := & HMAC_{hash}(secret, A_1 + seed) + \\ & HMAC_{hash}(secret, A_2 + seed) + \\ & HMAC_{hash}(secret, A_3 + seed) + \ldots, \end{aligned} \tag{8.1}$$

where $+$ denotes bit string concatenation, and

$$\begin{aligned} A_0 &:= seed \\ A_i &:= HMAC_{hash}(secret, A_{i-1}). \end{aligned}$$

So that the pseudo random function can be defined as

$$\text{PRF}(secret, label, seed) = P_{hash}(secret, label + seed), \tag{8.2}$$

where $label$ is an ASCII-encoded string, $seed$ a nonce[1], and $secret$ some (shared) secret.

All key-material used for later communication is derived from a master secret that is negotiated during the handshake protocol from exchanged pre-master-secrets, commonly Diffie-Hellman parameters. Pre-master secrets are either included in Certificate messages or explicitly exchanged using `ServerKeyExchange` and `ClientKeyExchange` messages.

48 bit source for keys

$$\begin{aligned} master_{secret} = & PRF(pre-master_{secret}, \text{master secret}, \\ & ClientHello.random + ServerHello.random) \end{aligned} \tag{8.3}$$

TLS Finish Message

The `Finish` message is a very crucial part for the protocol. It has to be exchanged after the exchange of `ChangeCipherSpec` messages. The messages contain the result of an application of the agreed upon pseudo-random function onto important parts of the protocol, including the master secret. This ensures, that a man-in-the-middle attacker is not able to exchange any of the messages of the handshake protocol in authenticated sessions, i. e., an attacker is not able to drop the communication onto the least-secure cipher suite.

$$PRF(master_{secret}, finishedlabel, Hash(handshakemessages))$$

8.3.2 SSL 2.0

The second version of the Secure Socket Layer protocol had security issues that make it advisable to cease using it. Currently the protocol is disabled by default in most browsers.

- Identical cryptographic keys are used for message authentication and encryption.
- weak MAC construction that uses the MD5 hash function with a secret prefix, making it vulnerable to length extension attacks.

[1] number used once, hopefully random

- no protection for the handshake, meaning a man-in-the-middle downgrade attack can go undetected.
- TCP connection close to indicate end of data. This means that truncation attacks are possible: the attacker simply forges a TCP FIN, leaving the recipient unaware of an illegitimate end of data message (SSL 3.0 fixes this problem by having an explicit closure alert).

8.3.3 StartTLS

Classically TLS is used on dedicated ports only. For example, HTTP is usually handled on port 80 and HTTP encapsulated in TLS is usually handled on port 443. We could argue, that this is a little impractical because now every service would require two distinct default port addresses, instead of one. Alas this is a historical artefact and will remain for the forseeable future, yet there is a different way of handling this.

StartTLS allows to "upgrade" any connection, e. g., a TCP connection, into an TLS-protected connection. StartTLS is commonly used for email protocols, notably SMTP and IMAP. The general concept is that the client, at the beginning of a communication session, issues a STARTTLS-command which then initiates the TLS handshake. (I am not sure whether also the server could issue a STARTTLS, in theory this is perfectly possible.)

- RFC 2817: HTTP Upgrade to TLS
- RFC 2818: HTTP Over TLS

8.4 Public Key Infrastructure (PKI)

How can one assume that a key is authentic?

A *Public Key Infrastructure* (PKI) denotes a hierarchical structure to issue, distribute and verify certificates.

Cross Certification means the mutual authorisation of certification authorities that expresses trust in the other CA and enables usage of signatures from one PKI tree within another. In practise this is implemented by providing a second certificate to a self-signed root-certificate.

Take for example two CA A and B who have established mutual trust. That means that CA A authorizes all certificates signed by CA B and vice versa. This would establish an alternative trust-paths for the certificates signed by any of the two CA.

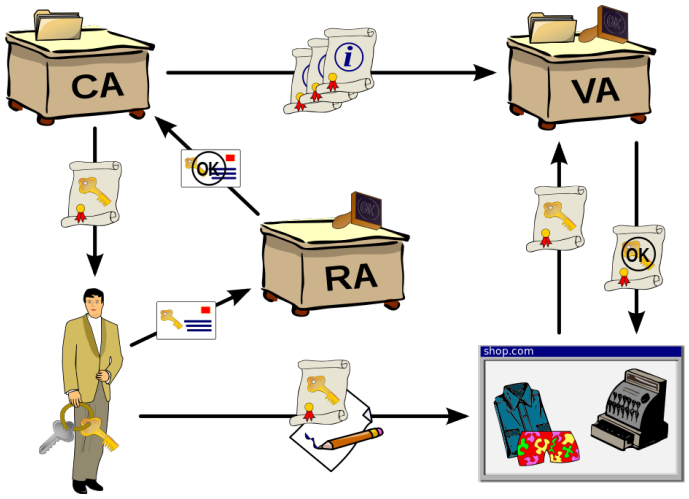


Figure 8.5: PKI Participants

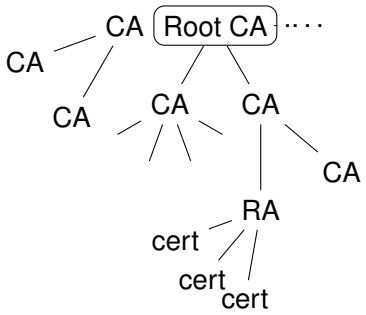
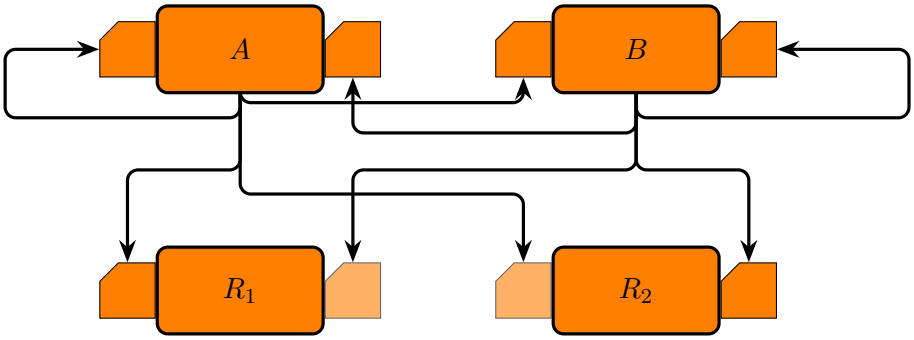


Figure 8.6: Tree of Certificates



X.509 Certificates

In this section we introduce the structure of X.509 certificates version 3. X.509 is an ITU-T[2] standard that defines a Public Key Infrastructure (PKI) and a Privilege Management Infrastructure (PMI). The PKI is expressed by certificates that bind identities to public keys by way of signatures of certification authorities (CA).

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (opt)
  - Subject Unique Identifier (opt)
  - Extensions (opt)
- Certificate Signature Algorithm

[2] International Telecommunication Union — Telecommunication Standardization Sector

- Certificate Signature

Certificate Revocation List

Certificate Revocation Lists (CRL) provide revocation of certificates in the X.509 PKI scheme. A revocation list is published by a CA and contains identifiers of certificates that became invalid before their validity time expired. CRL are standardised in RFC 5280.

There are various circumstances under which a certificate has to be revoked. Among others there are: Revocation vs. Hold

- Key Compromise The private key of the holder of a certificate has been disclosed to an attacker. Even the suspicion of a compromised key might be sufficient reason to order a re-certification.
- CA Compromise the issuing certification authority has been compromised and the currently signed certificates may have been faked. This obviously is a very big issue and there might be a lot of revocation certificates be circulating at the same time. Probably all certificates signed by this authority are affected.
- Privilege Withdrawn means that a privilege that has been granted by a certificate, for example the privilege to sign certificates, has been withdrawn from the holder — possibly a CA.

Critique of PKI

- Impersonal Trust Relation
- Difficult to Control Trustworthiness of CA
- Certificate Revocation is Slow and Blacklist-based
- Complicated Protocol for CRL/Cert. Distribution
- X.509 is complex
- X.500 Identifiers complex and not widely used

8.5 Key Exchange

9

Key Exchange Protocols

In this lecture, we will concern ourselves with the problem of distributing authentication and keys, especially session keys, between communication partners. This lecture is based on Chapter 9 from the textbook “IT-Sicherheit” by Claudia Eckert. We use this lecture to introduce some principles based on “Prudent Engineering Practices for Cryptographic Protocols” by Martin Abadi and Roger Needham. [**abadi96prudent**].

The introduced protocol is fundamental for authentication and communication security over un-secured networks in the Kerberos protocol. Kerberos itself is the default protocol for remote authentication in the Windows operating system, used to authenticate members of a service group. It is is also used in Unix-like operation systems, for example in the Andrew File System (AFS).

The underlying idea is, that a (central) authentication server establishes the authenticity of clients and provides session keys that authenticate the client against application servers while also securing the communication between clients and servers. The advantage, as compared to every server handling its own authentication, is, that the security checks are kept at one place, preventing that every server has to implement them itself.

The topic of this section must be distinguished from the common key-generation protocols, e. g., Diffie-Hellman in TLS, that are used to generate session keys. The protocols herein are more focussed on authentication by a single (often local) authority.

9.1 Key Exchange

Motivation for Key Exchange/Authentication:

- Ephemeral Session Keys

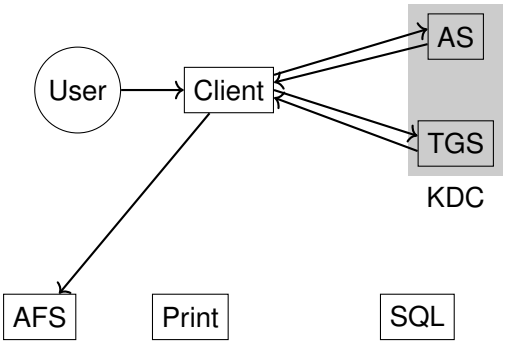


Figure 9.1: Kerberos Architecture

1. A : generates $K_{A,B}$
2. $A \longrightarrow B$: $\{K_{A,B}\}_{K_b}$
3. B : decrypts to $K_{A,B}$

Figure 9.2: Simplistic Protocol

  - Considered broken/disclosed after some usage
- Provide Perfect Forward Secrecy (PFS)
- Key Distribution
- Authentication of Participants

The canonic architecture of a Kerberos installation consists of the Key Distribution Centre (KDC) which itself consists of an Authentication Service (AS) and a Ticket Granting Service (TGS). A user wishing to use a service, e. g., the Andrews File System (AFS) first authenticates against his client, as usual by a password. The client then acts in the name of the user and requests an authentication ticket, i. e., a key to communicate with a service. To achieve this he has to authenticate its user against the AS and is then receiving a ticket from the TGS on request. This ticket then can be used to access the service. (See Figure 9.1.)

9.1.1 Simplistic Protocol

The most simplistic protocol (without actual authentication) is that all servers trust all clients A and simply accept key and user id from them.

The Simplistic Protocol (Figure 9.2) fails for various reasons.

- No authentication of Alice. Anybody could have generated the key. An Attacker could easily distribute its own key to Alice and Bob and then listen on the conversation. (This could be countered by explicitly stating

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: $\{K_b, B\}_{K_s^{-1}}$
3. $A \longrightarrow B$: $\{N_a, A\}_{K_b}$
4. $B \longrightarrow S$: B, A
5. $S \longrightarrow B$: $\{K_a, A\}_{K_s^{-1}}$
6. $B \longrightarrow A$: $\{N_a, N_b\}_{K_a}$
7. $A \longrightarrow B$: $\{N_b\}_{K_b}$

Figure 9.3: Needham-Schroeder Protocol with Asymmetric Keys

who initiates the conversation.)

- Man-in-the-Middle by intercepting the message and replacing it.
- Replay of key is possible. The session key could be cracked, given sufficient time, and then replayed back. (This would work even, if

9.2 Needham-Schroeder Asymmetric

The Needham and Schroeder are namesakes for two protocols for authentication and distribution of session keys.

Seven steps, described in [**Needham:1978:UEA:359657.359659**]

Problems with Needham-Schroeder:

- Missing Timelyness
- Replay of Authentication Messages (Nonces, see below)

9.3 Attack on Needham-Schroeder

The following attack is introduced by Gavin Lowe in [**Lowe95attackNeedhamSchroeder**]. It allows an attacker I to pose himself as A towards B.

Consider only the messages that facilitate authentication: 3, 6, and 7. An attacker I could now start an authentication with A and using this process to authenticate himself as A in a second conversation he starts with B:

where I is the attacker, denoted by $I(A)$ if posing as A, and the index number $i.j$ denotes protocol step j in the i-th conversation. (The first is undertaken with A and the second is undertaken with B.) The problem here is, that the messages do not make clear, for which communication the nonces are intended and thus do allow to swap them freely between different communications.

This flaw possibly motivates the statement of the third principle (Figure 9.8).

$$
\begin{array}{lrl}
1.3 & A \longrightarrow I: & \{N_a, A\}_{K_i} \\
2.3 & I(A) \longrightarrow B: & \{N_a, A\}_{K_b} \\
2.6 & B \longrightarrow I: & \{N_a, N_b\}_{K_i} \\
1.6 & I \longrightarrow A: & \{N_a, N_b\}_{K_a} \\
1.7 & A \longrightarrow I: & \{N_b\}_{K_i} \\
2.7 & I(A) \longrightarrow B: & \{N_b\}_{K_b},
\end{array}
$$

Figure 9.4: Attack on Needham-Schroeder Protocol with Asymmetric Keys

> Principle 3: If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message. [**abadi96prudent**]

Figure 9.5: Prudent Engineering Practice, Principle 3

9.3.1 Revised Protocol

Thus it is important to explicitly state the identity of partners, especially in the authenticating parts of the protocol.

A way to prevent the previously described attack is proposed in [**Lowe95attackNeedhamSchroeder**] Include responders identity in message 6:

6'. $B \longrightarrow A$: $\{B, N_a, N_b\}_{K_a}$ In that way the attacker is not able to just relay this message to A, who would expect a message from I, not from B. With the full revised protocol shown in Figure 9.6

9.3.2 Session Key

One Problem with the asymmetric scheme is, that there still is no session key established. And asymmetric encryption schemes are usually not known for their efficiency. There is, thus an eighth message to be send — at least — to establish $K_{a,b}$. Or, as has been explained in previous lectures, negotiated, for

$$
\begin{array}{lrl}
1. & A \longrightarrow S: & A, B \\
2. & S \longrightarrow A: & \{K_b, B\}_{K_s^{-1}} \\
3. & A \longrightarrow B: & \{N_a, A\}_{K_b} \\
4. & B \longrightarrow S: & B, A \\
5. & S \longrightarrow B: & \{K_a, A\}_{K_s^{-1}} \\
6'. & B \longrightarrow A: & \{B, N_a, N_b\}_{K_a} \\
7. & A \longrightarrow B: & \{N_b\}_{K_b}
\end{array}
$$

Figure 9.6: Revised Needham-Schroeder Protocol with Asymmetric Keys

1. $A \longrightarrow S$: A, B, N_a
2. $S \longrightarrow A$: $\{N_a, B, K_{a,b}, \{K_{a,b}, A\}_{K_{b,s}}\}_{K_{a,s}}$
3. $A \longrightarrow B$: $\{K_{a,b}, A\}_{K_{b,s}}$
4. $B \longrightarrow A$: $\{N_b\}_{K_{a,b}}$
5. $A \longrightarrow B$: $\{f(N_b)\}_{K_{a,b}}$

Figure 9.7: Attack on Needham-Schroeder Protocol with Symmetric Keys

example using the Diffie-Hellman protocol[1].

Simplistic: 8. $A \longrightarrow B$: $\{K_{a,b}, A\}_{K_b}$

With freshness: 8'. $A \longrightarrow B$: $\{\{K_{a,b}, A, B, T\}_{K_a^{-1}}, A\}_{K_b}$

Clever: 8". $A \longrightarrow B$: $g^a mod n$
8". $B \longrightarrow A$: $g^b mod n$ (Diffie-Hellman)

9.4 Needham-Schroeder Symmetric

This protocol utilises a trusted server S to authenticate two participants A, B and securely generate and distribute a session key. We will successively learn, that this first version of the protocol thus had its flaws and had to be revised, before it could be incorporated into real-world protocols.

The protocol (Figure 9.10) runs as follows. A contacts the server, expressing the wish to communicate secure and authentic with B. The server S then answers with an message encrypted with the master-key between A and S. The included nonce N_a provides freshness and thus protection from replay attacks. $K_{a,b}$ is the session key for communication with B, which is included twice: directly, readable for Alice and within an enclosed data packet, which is encrypted with the master secret $K_{b,s}$.

The enclosed packet additionally contains the identification of Alice to state, that the enclosed key is usable only in communication between Alice and Bob. This enclosed packet is then relayed by Alice to Bob.

- Symmetric is faster
- Shorter Key Length
- Session Key are symmetric too

At that point Alice can expect that only Bob is able to use $K_{a,b}$. But Bob has yet no proof that Alice is authentic. The next two messages are used to authenticate Alice. Bob crafts a nonce, and sends it encrypted to Alice. Alice, if

[1] But keep in mind, that Diffie-Hellman was only two years published at that time and thus quite uncommon, as was public-key cryptography.

> Principle 3: If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message. [**abadi96prudent**]

Figure 9.8: Prudent Engineering Practice, Principle 3

authentic, is able to decrypt the nonce, apply a simple transformation that is known by both parties to it and sends it back using the session key again.

In the symmetric protocol encryption is used both for secrecy of the session key and mutual authentication of clients. If Alice is able to use $K_{a,b}$, then Bob assumes her authenticity and vice versa. Alice on the other hand has to trust S, that it encrypted the key in a way that only Bob is able to use it.

The following principle is not wronged by the use in the symmetric protocol, as long as we are clear about the meaning of encryption.

9.4.1 Timeliness of Communication

The problem of the protocol is, that it assumes, that no session key is ever disclosed to an attacker, or cracked.

But having gained knowledge on any session key, an attacker could simply replay the third message and would be able to correctly answer the challenge within the fourth message.

3' $I \longrightarrow B$: $\{K_{a,b}, A\}_{K_b}$
4' $B \longrightarrow I$: $\{N'_b\}_{K_{a,b}}$
5' $I \longrightarrow B$: $\{f(N'_b)\}_{K_{a,b}}$

Thus, the attacker would be able to pose as Alice, because an old session key has been compromised. This is exactly not what Perfect Forward Secrecy is about. The reason is, that we have attached too much meaning to the nonce N_b, which proves knowledge of the key, but can not guarantee, that this is a recent key. Which motivates the next principle:

> Principle 6: Be clear what properties you are assuming about nonces. What may do for ensuring temporal succession may not do for ensuring association and perhaps association is best established by other means. [**abadi96prudent**]

Nonces itself are not generally sufficient to guarantee that messages are fresh and not simply replays. The attribute that the nonces in the above protocol shall prove is knowledge of a private key (via the ability to decrypt a nonce) but this does not prove freshness.

1. $A \longrightarrow S$: A, B, N_a
2'. $S \longrightarrow A$: $\{N_a, B, K_{a,b}, T, \{K_{a,b}, A, T\}_{K_{b,s}}\}_{K_{a,s}}$
3'. $A \longrightarrow B$: $\{K_{a,b}, A.T\}_{K_{b,s}}$
4. $B \longrightarrow A$: $\{N_b\}_{K_{a,b}}$
5. $A \longrightarrow B$: $\{f(N_b)\}_{K_{a,b}}$

Figure 9.9: Attack on Needham-Schroeder Protocol with Symmetric Keys

0.1 $A \longrightarrow B$: A
0.2 $B \longrightarrow A$: $\{A, N_0\}_{K_{b,s}}$
1. $A \longrightarrow S$: A, B, N_a
2'. $S \longrightarrow A$: $\{N_a, B, K_{a,b}, \{K_{a,b}, A\}_{K_{b,s}}, N_0\}_{K_{a,s}}$
3. $A \longrightarrow B$: $\{K_{a,b}, A, N_0\}_{K_{b,s}}$
4. $B \longrightarrow A$: $\{N_b\}_{K_{a,b}}$
5. $A \longrightarrow B$: $\{f(N_b)\}_{K_{a,b}}$

Figure 9.10: Attack on Needham-Schroeder Protocol with Symmetric Keys and prior nonce

9.4.2 Revised Protocol (2)

Thus, a second way to improve the original Needham-Schroeder Protocol is to not only be explicit on the intended partners of the communication, but to provide freshness as well.

The easiest way is to introduce time stamps, which thus would require synchronised time between communication partners.

Another way is to exchange nonce directly before communicating to the server. This could be seen as a commitment on a given time.

9.5 Certificate-based Key Exchange

What is the difference between Key Exchange Protocols and PKI infrastructures, i.e. certificates? Why not simply all use x509?

Key Exchange

- Interaction based
- Current communication
- Establish Session Key

Certificates

- Asynchronous

- Revocation difficult Either defined validity time, or certificate revocation lists, or both.

9

Digital Money

Contents

Lernziele:

- Zero-Trust Scenarios
- Homomorphic Encryption
- Distributed Ledger Technologies
- Bitcoin Transactions
- Hash Cash
- Blockchain

9.1 Overview

Most — and in practice probably: all — electronic currency transfers currently are facilitated by shifting values in account books (or ledgers) and have little similarity with cash currencies. Cash and ledgers have very different properties although both share the functionality of providing control over defined values. The main difference is in the way how *control* is established and *access* is protected, but mainly in the *traceability* of transactions.

9.1.1 Digital Payment

The base idea of money is to provide a universal value exchange so you don't have to concern yourself with the hassles of a swap-market, e. g., swapping corn for tools. The money-concept has been implemented using various physical types of tokens; metal coins, paper slips, and, among many others even immobile stones[1].

E-payments are payments made over the internet[. . .]

[1] Technically all "coins" of the Yapese stone money could be moved, but with a diameter of more than 3 meters that arguably would require some large pockets.

1) Remote payment card transaction through the internet.
2) Online-banking based credit transfers or direct debits under which the payer uses an online banking portal for authentication (currently only operational at domestic level).
3) Payments through e-payment providers, with which the consumer has set up an individual account. Accounts can be funded through 'traditional' payment methods, for example bank transfers or credit card payments.

m-payment, proximity payment e. g., NFC devices

definitions related to money

[http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2011:0941:FIN:EN:HTML, 2013-06-13]

Cryptocurrency

E-Money

- Ledger Based/Bank Accounts
  - Default Bank Transaction
  - Traceability
  - Monolithic Control
  - Weak Trust Links
  - Homebanking Computer Interface (HBCI, FinTS 4.0)
- Payment, Debit, Credit. . . Cards

9.1.2 Digital Money

It would be very nice to have something that resembles cash money, only it should be represented in a digital way, so that we could pay using digital devices in a similar way we use our wallets. Digital money than could have additional advantages most craved the ability to be transferred directly.

> "*Electronic money* is a digital equivalent of cash, stored on an electronic device or remotely at a server." [http://ec.europa.eu/internal_market/payments/emoney/, 2013-06-13]

- Electronic purse (stored locally)
- Micro Payments
- Smart/Payment Card, Cell phone

But creating a cash-equivalent electronic payment system that mimics all attributes of cash money, while allowing secure and easily usable payment over communication networks has proven to be a difficult task. Notable advances have been undertaken, most famously, by David Chaum in his work in the

1980s. [**chaum1983:blindsignaturesuntraceable**] [**Chaum1985SecuritywithoutIdentification:**] [**chaum1988:untraceableelectroniccash**]

Attributes of Money

- Distributed storage
- Local transfer (Spatially close)
- No posterior traceability (needs preparation)
- Anonymity of owner (no traces on money)
- Amount owned is "secret"
- Expensive forgery prevention/expensive forgery

Already Wikipedia knows of 34 different cryptographically protected electronic currency systems, of which 23 are considered active. The most popular and the root of most of them seems to be Bitcoin[2], which will be covered in Section 9.4.

- relying on cryptography
- mostly Proof-of-Work schemes
- often peer-to-peer, decentralised
- no inflation (often)
- protection against frozen values

9.2 Chaum's Digital Cash

David Chaum's famous approach to creating a digital currency is actually based on an attack against RSA. Davida's Attack exploits the homomorph structure of RSA.

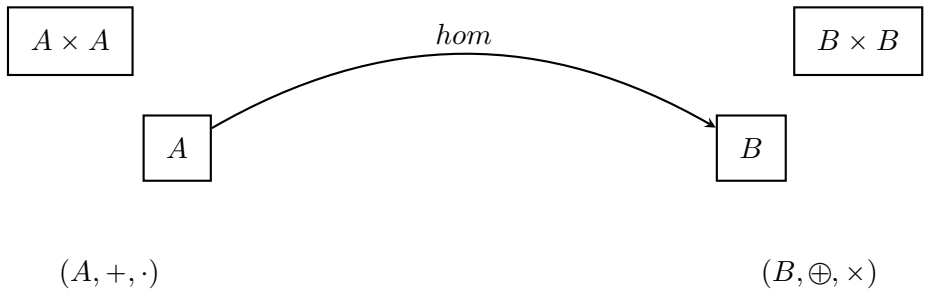
9.2.1 Davida's Attack on RSA

RSA is a partially homomorphic encryption algorithm. This remarkable feature provides that certain operations can be executed on *encrypted* data, without decrypting the data beforehand. If the algorithm is homomorphic, than this operation is equivalent to an operation on the decrypted data. This, homomorphic encryption algorithms allow mathematical operations on data, without being able to decipher the data[3]

[2] `http://bitcoin.org`
[3] pretty magical, ain't it?

A homomorphism is a mapping between the elements of two algebraic structures of the same typ. Meaning both structures provide a similar set of operations (with same arity) on their respective sets.

Sets A, B and operations $+, \cdot, \oplus, \times$.



e. g., $\forall x, y \in A : hom(a) \times hom(b) = hom(a \cdot b)$

Homomorphisms can be understood as maps that retain the essential structure of the preimage within the target domain. If an inverse homomorphism $hom^{-}1$ exists, both structures are called *isomorph* and the homomorphism is called an *isomorphism*.

The homomorphism in RSA allows for homomorphic multiplication. Thus, you could multiply on encrypted data. The only requirement is, that the data be encrypted with the same RSA-key pair.

$$E(x) \cdot E(y) = E(x \cdot y) \tag{9.1}$$

Now, homomorphic encryption can be used to delegate computations on sensible data, without disclosing that data.

In [**davida1982chosen**] utilises the RSA homomorphism to manipulate a principal into signing an encrypted message. If the same public/private pair of keys is used for both encryption and signature, then the application of the signature reverses the encryption. In order to hide this from the signer, the encrypted message is “blinded” beforehand.

Davida’s attack not only teaches us a conceptual vulnerability of asymmetric cryptography but also the very useful technique of blinding, which can also be used for blind signatures in electronic money schemes and anonymous credentials.

In the first step the attacker intercepts a message m encrypted with the public key e of the victim. The attacker then multiplies an arbitrary value y encrypted

for the victim.

$$
\begin{array}{rcl}
\text{given: } m^e & & \\
\text{select: } y & \longrightarrow & x = y^e \\
\text{blind: } m^e & \longrightarrow & x \cdot m^e \\
\text{victim signs: } x \cdot m^e & \longrightarrow & (x \cdot m^e)^d \\
(c^{\cdot e})^d & \overset{\text{RSA Hom.}}{\Longrightarrow} & x^d \cdot m^{e \cdot d} \\
 & & = y^{e \cdot d} \cdot m^{e \cdot d} \\
 & & = y \cdot m \\
\text{unblind: } y \cdot m & \overset{/y}{\Longrightarrow} & m
\end{array}
$$

The combined value of encrypted message and value is the forwarded and the victim is foiled into signing the value, e. g., by using it as part of another statement. By the RSA homomorphism this is the same as applying the private key individually to the encrypted message and the encrypted, attached value.

The result is the product of attached value and message. As the value has been created by the attacker, he is able to extract the unencryptey message by dividing through y.

This provides us with two general rules on the handling of asymmetric keys:

1. Never use the same key-pair for encryption and signature, and
2. Be clear about what the meaning of a signature of a given document is.
3. Do not sign messages that you cannot read/understand.

9.2.2 Chaums Bank Signature

The RSA homomorphism can, in return, be used very elegantly to provide untraceable digital money [**chaum1983:blindsignaturesuntraceable**]. The objective is to generate some equivalent to coins that have a fixed value and whose authenticity can be evaluated easily by everyone. Furthermore should neither the issuer of the money (e. g., a bank), nor the dealer where the money is spend be able to link the money to a given customer's account.

The coins of the money consist of a value w that is specially formed in an easily recognised structure, here we use duplication, i. e., concatenation of two equal numbers denoted $v.v$. The value w is signed by a special key that provides the monetary value, e. g., the signature key-pair s, t is the private/public signature key pair used by the bank to sign "5€". That means, the value of the money is not denoted by w, but by the signature key used by the bank.

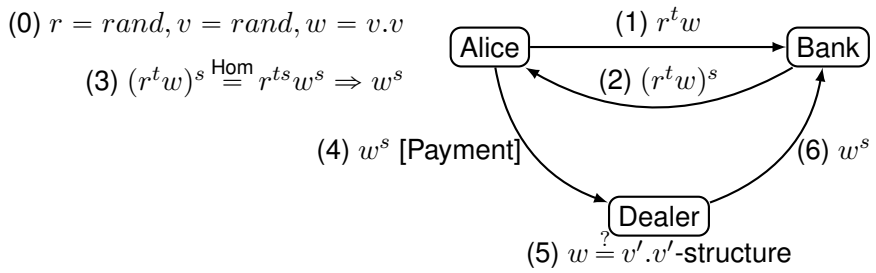


Figure 9.1: Chaum's Digital Cash — Messages and Calculations

The customer (Alice) generates w and "encrypts" a random number (called *blinding factor*) using the public verification key of the bank related to the intended monetary value. The product of blinding factor r^t and doubled structure w is send in an authentication communication to the bank. The bank signs the product with the matching private signature key s, returns the signed value to Alice and debits the account of Alice with the fitting amount.

Alice can now utilise the RSA homomorphism to derive a coin, i. e., the signed double structure w^s.

To pay Alice simply has to transmit the coin to a dealer who can verify the signature by applying the fitting public verification key t, which must reveal the special structure of w. To prevent double spending of the coin, the dealer immediately has to transfer the coin to the bank. The bank has to store a history of all spent coins and verify that any received coin is no attempt at double spending. After verification of the signature and clearance from the coin history, the bank can transfer the assigned amount to the dealer's account.

During communication with Alice, the bank could gain no knowledge about w, due to the unknown blinding factor. The bank thus is not able to relate the communication with Alice to the communication with the dealer. The dealer only can infer that Alice has an account with the bank, although it can be imagined that proxy money brokers may hide this relation.

The main drawback of the scheme is, that every transaction has to be cleared immediately with the bank. And that the bank needs to store a history of spend coins. This obviously leaves the customer with the risk of generating, by chance, a spend coin. As this cannot be verified when the coin is "minted" by the bank, a customer always has a chance of generating a duplicate of a coin. This motivates a behaviour of drawing ecash only briefly before the money is spend, which creates relations between the minting process and the spending of a coin.

9.3 HashCash

Developed by Adam Back in 1997 to limit Spam and Denial-of-Service attacks. The idea was, that the originator of an email would need to provide proof that he had spend at least a minimum of computing power to generate a special header for this message. [**Back02HashcashDenialof**] It further is fundamental for the understanding of Bitcoin, which is discussed in Section 9.4.

- Adam Back 1997
- Objective: prevent Spam and DoS
- Idea: force expensive computation by sender
- Requirement: inexpensive validation by recipient
- Method: brute-force computation for specially formed hash value

This special *header* would consist of a random number, the date-time, an identifier of the sender — e. g., an email address — and a hash of these fields. The hash, thus, has a special requirement imposed, which is, that a certain number of bits at the beginning has to be zero bits. If the calculated hash does not have this attribute, the sender has to increment the random number until he finds a hash that has.

$\underbrace{000000000000}_{n \text{ Zero-bits}}$f3ff35db5f58cba4a56ac57c9239

Computation effort is

Exponential to the number of required zero-bits.

Computation of a well-formed hash requires a brute-force enumeration by repeated increment of the counter (or otherwise guessing input variables) until a well-formed hash is produced. Since on average every 2^nth hash produces a value with n leading zeroes, it takes that many hash value computations on average until a well-formed hash is produced.

HashCash itself used an 160 bit SHA-1 hashing algorithm.

Sender:

- Write Message
- Do Work and Generate Proof-of-Work (PoW)
- Append PoW to Message

Receiver:

- Validate PoW
- Accept Message

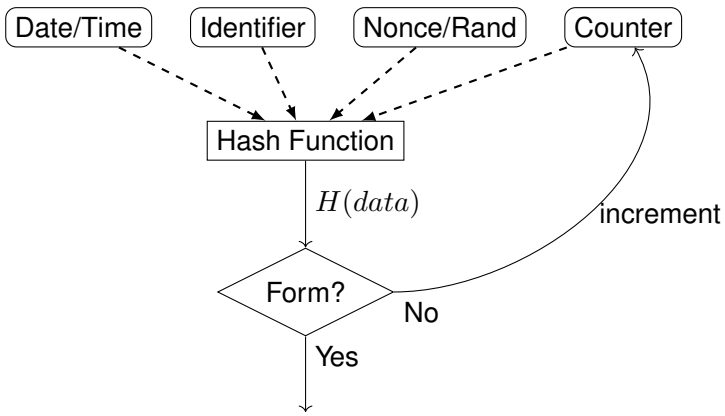


Figure 9.2: HashCash Principle

In greater detail, a HashCash v1 header consists of 7 fields[4]:

`1:bits:date:resource:ext:salt:suffix`

1 The version number (version zero is simpler, but has some limitations).

bits Number of leading zeroes. Denoted *bit value*. If the stamp does not really hash with the purported leading zero bits, it is not valid.

date date (and time) a stamp was minted. Stamps in the future and those too far in the past may be judged invalid.

resource An identification of the resource for which the stamp was minted. Perhaps an e-mail address, but also possibly a URI or other named resource.

I am curious, why nobody mentiones the recipient address too, I find it missing here.

ext Any additional data Extensions that a specialized application may want. Any additional data could be placed in here, but in usage so far, this field is generally left empty.

salt A random salt that distinguished this stamp from any other one minted for the same resource and date. For example, two different people may reasonably want to send e-mail to my same address on the same day. They should not be disqualified by my use of a double spend database. But if each of them uses a random salt, their complete stamps will differ.

suffix The incremented value. The suffix is the real work of the algorithm. Given the first six fields, a minter must try many sequential suffix values to produce a stamp that hashes with the desired number of leading zeros.

The recipient now simply has to validate the hash, verify that it meets the spe-

[4](I quote freely from `http://www.ibm.com/developerworks/linux/library/l-hashcash/index.html` here.)

cial requirement, check whether the message is sufficiently fresh[5], and compare the identifier contained in the header with the expected identifier.

The reason, why this scheme works to prevent spam, while still being usable, is that for spamming to be efficient it needs a large throughput of multiple messages per second. As the computation of a single hashcash takes up to a few seconds, the computation becomes forbidding complex for the spammer, reducing his throughput up to only one spam every few seconds. The reception of a spam is not hindered. But there still is a problem with this idea.

HashCash has been controversially discussed. In 2004 Laurie and Clayton showed, that spam is not sufficiently hindered by HashCash in a workshop contribution to the Workshop on the Economics of Internet Security. [**LC04ProofofWork**]. The HashCash system is (allegedly) already implemented in a number of email-clients, spam filters and web applications (blogs).

The underlying assumption of HashCash is, that the perpetrator, the spammer, is the one forced to carry the burden of Proof-of-Work. But that is, in times where spam is mostly send using extensive bot networks, a difficult assumption. As spammers are utilising computers of unsuspecting individuals for computation, the protection by HashCash is nullified.

Another problem is with mailing-list servers. Obviously a server that has to re-calculate a HashCash for every outgoing e-mail would need much more resources than the original sender of the message. This obviously provides a great opportunity for Denial-of-Service attack, making mailing-list servers profoundly vulnerable. This obviously is not a solution.

9.4 Bitcoin

Bitcoin [**nakamoto2008bitcoin**] is a ledger-based system where copies of the account book are stored on all participant's computers. That means, that every transaction is published (and eventually recognised) to all participants. Although Bitcoin is based on a global account book, it is possible to make offline payments.

9.4.1 Bitcoin History

Following is a very brief selection of Bitcoin historic events:

2008 Design Paper, Domain Name, Sourceforge

2009 Genesis Block, First Transaction

[5] i. e., the date-time is recent

2010 v0.3, First Pizza bought, MtGox established, $0.50/BTC, total economy $> \$1$ million microtransaction exploit

2011 1st quarter of BTC mined, $1,...$10/BTC, BTC Client exploit (CVE-2011-4447)

2012 Bitcoin Foundation, BTC exchange "bank" in europe

2013 BTC client v0.8, market cap $> \$1$ billion, stable >$100/BTC

[https://en.bitcoin.it/wiki/History]

9.4.2 Bitcoin-based Cryptocurrencies

Currently there is not only Bitcoin, but a (growing) list of electronic currencies that work by similar concepts as Bitcoin, i. e., securing transactions in block chains, using proof-of-work for fraud protection and generating money from proven work. You find a list of *notable* cryptocurrencies at wikipedia[6], which mentions around 60 existing currencies in December 2013.

Bitcoin and related currencies seem to gain importance. The trading value of the coin itself has just recently[7] risen as is shown in Figure 9.3. And the currency is already not only usable for trading contraband but there is an evolving marketplace for almost any good[8].

9.4.3 Bitcoin Market

Bitcoin nowadays can be traded similar to conventionel currencies. It thus undergoes similar fluctuations. Figure 9.3 shows the Bitcoin hype in 2013-05 which was probably related to the banking crysis of 2013. The current maximum market value of roughly USD 1000 for a single Bitcoin has since then dropped slowly down to USD 450 which is still far above its value at the beginning of 2013. (See Figure **??**.)

9.4.4 Ecological Impact

9.4.5 Bitcoin Objectives

Objectives of Bitcoin are:

- Electronic Payment
- Based on cryptographic proof NOT trust
- Direct transaction (withouf TTP)

[6] https://en.wikipedia.org/wiki/Cryptocurrency, 2014-04-30
[7] Spring 2013
[8] https://en.bitcoin.it/wiki/Trade, 2013-06-03

Figure 9.3: Bitcoin Market Value (source: `http://blockchain.info/de/charts/market-price`, 2013-05-03)

- Prevent Double-Spending

Assume: Honest nodes control CPU $>$ dishonest nodes CPU

The biggest problem of every electronic currency is to prevent double-spending. For this, the rule is, that if two different transactions are tried on the same coin (by the same owner), only the first one in the global ledger counts.

And quite obviously the double-spender is a cheater.

Bitcoin shall work according to its specifications under the assumption that honest players in this scheme control more CPU power than cooperating dishonest players.

9.4.6 Bitcoin Blockchain

The central structure in Bitcoin is the block chain mechanism. A block chain is a tree of transaction blocks, starting at a *genesis block*. In Bitcoin the *main chain* is the longest series of blocks from the genesis block which defines the current valid chain of transaction blocks.

A blockchain is made of blocks that are “chained” by including a hash value of each block’s preceding block. A block collects a set of transactions, uniqueness of the block is provided by a nonce. And the integrity of a block can be protected via signature, but the final acceptance of a block as authentic is only given if a hash of that block is included in the (or any of the) following blocks. Thus, your block becomes valid by other principals accepting and using your block. (See Figure 9.5)

The hash value used in a block is where the actual proof of work is done. (See Section on HashCash). In the Bitcoin scheme the complexity of the generation of blocks is increasing with the length of the blockchain. This way it shall be ensured that the complexity of the Proof of Work keeps up with improvements

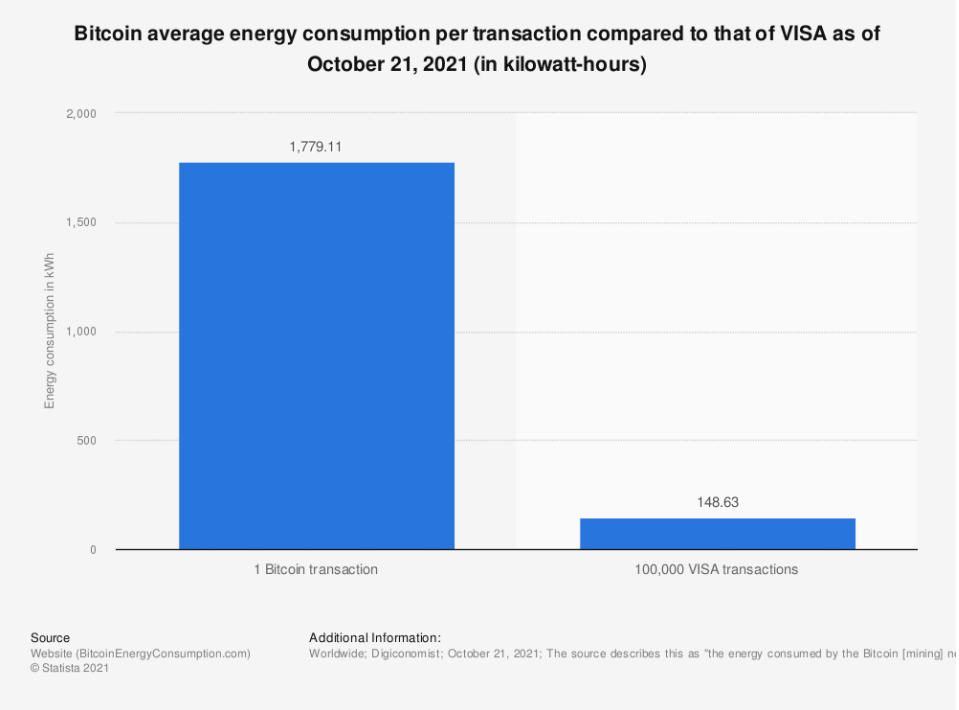


Figure 9.4: Energy consumption of a single Bitcoin transaction compared to 100,000 VISA transactions.

in computing power.

Thus, “a” blockchain can actually be a tree rather than a single chain, with each leaf-node possibly representing a different set of valid transaction. Obviously it is not beneficial if the global Bitcoin-ledger exists in multiple versions thus, the convention is, that the longest valid chain is considered to contain the representative ledger. As a chain requires the calculation of a Proof of Work for each block that is appended, the end of the chain which is worked on by the largest amount of computing power will grow fastest.

Assuming that no single entity is controlling 50% or more of the involved com-

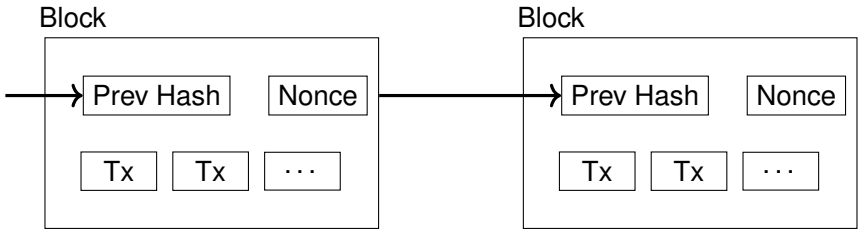


Figure 9.5: Bitcoin Transaction Blocks

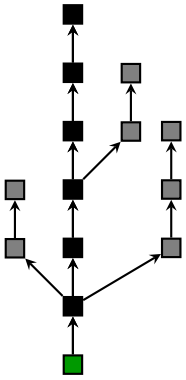


Figure 9.6: Block Chain (Green: Genesis Block, Black: Main Chain)

fix chain image, connect loose roots

puter resources, the decision on which chain is valid is approximately decided by a majority of blockchain miners. (See Figure 9.6)

Problems arise if equally powerful parts of the community are in dispute of the validity of different sets of transactions, or communication is separated. But, in the past these disputes and netsplits have been solved after some time.

- Transactions are broadcasted
- Each node collects transactions in a single block
- Each node works on finding proof-of-work for this block
- If found, proof-of-work is broadcasted
- Node accept block $\iff$ all transactions are valid (especially not spent)
- Block acceptance is expressed by working on the next block of chain (starting at step 2.)

The longest chain always wins.

9.4.7 Bitcoin Transactions

For this to work, a scheme that allows for participants to agree on a single history of transactions is needed.

Bitcoin Transaction Mechanics

A transaction consists of Inputs and Outputs, which ideally are summing up to the same amount. A transaction specifies how the amount inserted by the Inputs is distributed to different Outputs.

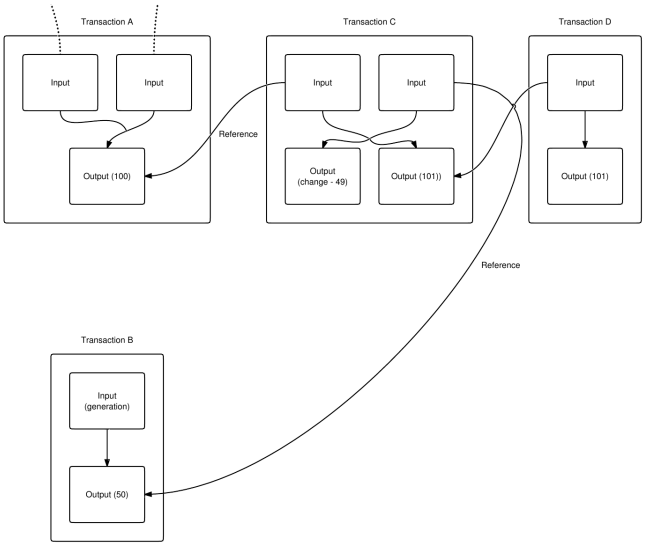


Figure 9.7: Example Transaction (`https://en.bitcoin.it/wiki/File:Transaction.png`, 2013-06-19)

Inputs are always referring backwards to individual Outputs from different transactions that already have happened. There thus is a one-on-one mapping from Inputs backwards to already existing Outputs.

Outputs do specify a represented amount and a recipient of that amount. There, though, is the possibility to generate transactions without explicit recipient.

Input reference to Txout, (Reference, Signature) Parts of a transaction where the signer receives money.

Output sending Bitcoins, (Value, PubKey) Parts where the signer sends/assignes money towards someone referenced by a public key.

The actual balance of an account is never explicitly included in a transaction or block, but only calculated from following all incoming and outgoing transactions of an account.

Figure 9.7 provides an example of the transactions that could be included in a single block: “A sends 100 BTC to C and C generates 50 BTC. C sends 101 BTC to D, and he needs to send himself some change. D sends the 101 BTC to someone else, but they haven’t redeemed it yet. Only D’s output and C’s change are capable of being spent in the current state.” [`https://en.bitcoin.it/wiki/Transactions`, 2013-06-19]

The general scheme is that every transaction is signed by the original owner of a given coin. The ledger forms a hash chain by combining the the current transaction with the previous transaction in a single hash, which is then signed

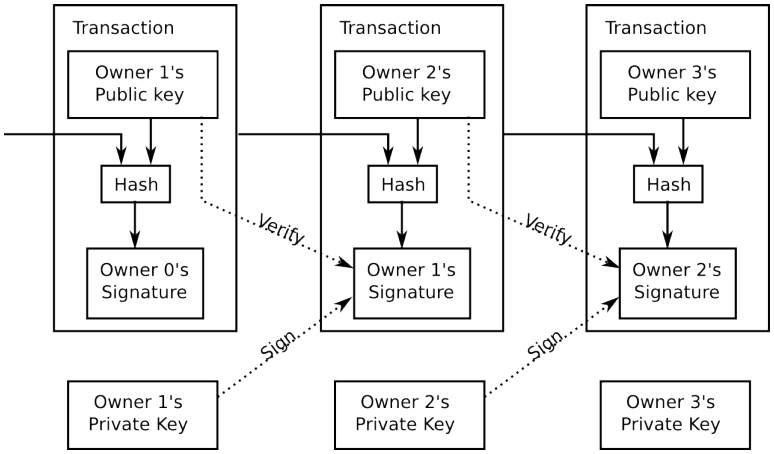


Figure 9.8: Bitcoin Transaction

by the private key. The hash is formed similar to the HashCash concept in a way that provides a Proof-of-Work to makes it hard to create fake histories.

9.4.8 Splitting and Combining Values

Every output in an transaction must be paired to inputs that provide at least the output's amount summed together. Every amount from the inputs that is not claimed in an output is considered an transaction fee and can/will be claimed by the user providing the Proof-of-Work for the encapsulating block. So, isn't that very much inconvenient, do I have to wait for another transaction, to claim change? And wouldn't this be insecure, because receiving change would be purely based on trust in the one I am just handing over all these Bitcoins?

Well, it actually is much easier than with cash, because you are always able to pay a fitting amount and directly reclaim the change in the denomination you find fitting. No more fishing for the right coins in your purse, you just stamp the right amount directly.

How do you do that? We already know, that a single transaction block may contain multiple inputs and outputs. The only thing you have to do is to generate outputs that can be redeemed by yourself.

Take, for example the transaction in Figure 9.9. Assume Out a is a payment to someone, that is backed by the Inputs. As the sum of amounts in the Inputs is alledgedly larger than the amount in Out a, the user is assigning the remainder back to himself in a second Output. Any amount that is not covered in an explicit Output is considered to be a transaction fee and may be claimed by the client providing the Proof-of-Work for the containing block.

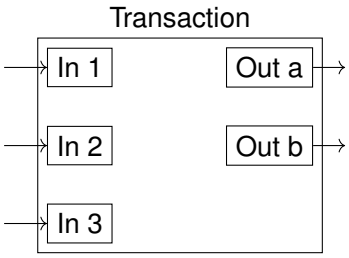


Figure 9.9: Splitting and Combining by Multiple In- and Outputs [**nakamoto2008bitcoin**]

9.4.9 Transaction Format

To understand transactions we take a brief look at the different protocol fields that make up a transaction. A transaction consists, as Figure 9.7 shows, of inputs and outputs.

Version (4)

#Txin (VI) Number of Inputs. Data format is a variable integer, length is 1-9 bytes

Txin i ($\geqslant 42$) i = 1,. . . ,#Txin Sequence of Inputs

#Txout (VI)

Txout i ($\geqslant 9$) i = 1,. . . ,#Txout

locktime (4)

Where (i) denotes the width of the field in bytes.

The following fields describe the Transaction Input Format:

Prev. Transaction Hash (32) SHA256

Prev. Txout-index (4)

Txin-script length (VI)

Txin-script/scriptSig

sequence no (4)

The Transaction Output Format consists of the fields:

Value (8) value in $1e^{-8}$ BTC

Txout-script length (VI)

Txout-script/scriptPupKey

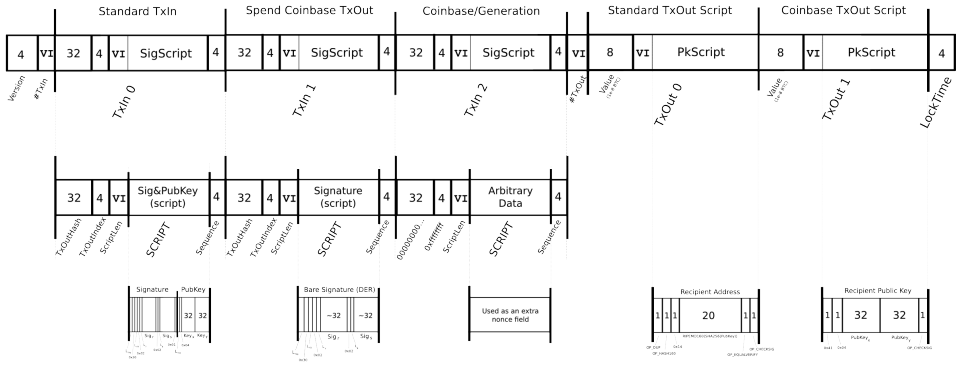


Figure 9.10: Bitcoin Transaction BinaryMap [Excerpt from `https://en.bitcoin.it/w/images/en/e/e1/TxBinaryMap.png`, 2013-06-19, by `etotheipi@gmail.com/1Gffm7LKXcNFPrtxy6yF4jBoe5rVka4sn1`]

9.4.10 Transaction Script

Bitcoin uses a special script language based to describe transactions. Because the way in which transactions can be handled is very flexible. This flexibility has allowed for the encoding of a wider variety of formal contracts, colloquially known as *Smart Contracts*. The Bitcoin Wiki provides a few examples.

- Loop-free, stack machine
- Binary encoded
- Constants, Comparision, Arithmethic
- Flow-Control (`OP_IF`,...)
- Stack-Operations (`OP_DUP`, `OP_DROP`, ...)
- Crypto
  - `OP_SHA256`: $in \rightarrow hash$ Takes input from the stack and returns a hash.
  - `OP_CHECKSIG`: $sigpubKey \rightarrow$ `true`/`false` Takes a Signature and a public key from the stack and pushes true onto the stack if both match to each other.

[see `https://en.bitcoin.it/wiki/Script`]

The transaction values shown in Figure 9.11 provide the base information needed in a transaction block. Within the input transaction the value of an the output transaction with index 0 of a previous transaction is claimed by providing a signature on the script of the referenced output transaction. (This is the input to the output script.)

Figure 9.11: Input and Output Transaction (Values of Fields)[https://en.bitcoin.it/wiki/Transaction, 2013-06-04]

The output transaction provides a value of the transaction and the recipient by defining the hash of the receiving public key. To claim this value the recipient must provide a signature on the script and the related public key, which must match the hash given in the script.

Figure 9.12 shows the first transaction in Block 170 consisting of one input and two outputs. The outputs are made directly to

Tbd. explain the fields in Bitcoin Block (Figure **??**)

Excursus: Merkle Trees

Merkle Trees are binary trees whose paths are hash chains. The root of the tree is the hash of two sub-trees. A sub-tree either is, again, the hash of two concatenated sub-trees or the hash of some data. The tree is constructed from the leaves on and thus well balanced. Please note, that the example in Figure 9.14 shows how odd numbers of leaves are handled by repeating the last leave.

Transaction Types

Bitcoin allows for three types of transaction. Bitcoins can be transferred to a recipient identified by its IP-address, identified by its Bitcoin identity. The third way of transfer is the generation of new money.

Transfer to IP address Simple Authentication by PubKey

Transfer to Bitcoin address Authentication by Recipient Hash

Generation Draw from *coinbase*

(Contract)

Transfer to IP

Input to Verification:

scriptPubKey: `<pubKey> OP_CHECKSIG`

scriptSig: `<sig>`

To facilitate “Transfer to IP” Figure 9.15 the script from the Input and its related Output are concatenated and executed by pushing constants to the stack (Step

```
{
"hash":"f4184fc596403b9d638783cf57adfe4c75c605f6356fbc91338530e9831e9e16",
"ver":1,
"vin_sz":1,
"vout_sz":2,
"lock_time":0,
"size":275,
"in":[
  {
    "prev_out":{
      "hash":"0437cd7f8525ceed2324359c2d0ba26006d92d856a9c20fa0241106ee5a597c9",
      "n":0
    },
    "scriptSig":"304402204e45e16932b8af514961a1d3a1a25fdf3f4f7732e9d624c6c61548
                 ab5fb8cd410220181522ec8eca07de4860a4acdd12909d831cc56cbbac4622
                 082221a8768d1d0901"
  }
],
"out":[
  {
    "value":"10.00000000",
    "scriptPubKey":"04ae1a62fe09c5f51b13905f07f06b99a2f7159b2225f374cd378d71302f
                    a28414e7aab37397f554a7df5f142c21c1b7303b8a0626f1baded5c72a70
                    4f7e6cd84c OP_CHECKSIG"
  },
  {
    "value":"40.00000000",
    "scriptPubKey":"0411db93e1dcdb8a016b49840f8c53bc1eb68a382e97b1482ecad7b148a69
                    09a5cb2e0eaddfb84ccf9744464f82e160bfa9b8b64f9d4c03f999b8643f6
                    56b412a3 OP_CHECKSIG"
  }
]
}
```


Figure 9.12: Bitcoin Transaction (First Transaction in Block 170) [`https://blockexplorer.com/rawtx/`, 2013-06-03]

```
{
"hash":"000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
"ver":1,
"prev_block":"0000000000000000000000000000000000000000000000000000000000000000
"mrkl_root":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33
"time":1231006505,
"bits":486604799,
"nonce":2083236893,
"n_tx":1,
"size":285,
"tx":[
  {
    "hash":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
    "ver":1,
    "vin_sz":1,
    "vout_sz":1,
    "lock_time":0,
    "size":204,
    "in":[
      {
        "prev_out":{
          "hash":"0000000000000000000000000000000000000000000000000000000000000000
                  0000",
          "n":4294967295
        },
        "coinbase":"04ffff001d0104455468652054696d65732030332f4a616e2f323030
                    204368616e63656c6c6f72206f6e206272696e6b206f66207365636f
                    64206261696c6f757420666f722062616e6b73"
      }
    ],
    "out":[
      {
        "value":"50.00000000",
        "scriptPubKey":"04678afdb0fe5548271967f1a67130b7105cd6a828e03909a679
                        e0ea1f61deb649f6bc3f4cef38c4f35504e51ec112de5c384df7
                        0b8d578a4c702b6bf11d5f OP_CHECKSIG"
      }
    ]
  }
],
"mrkl_tree":[
  "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
]
}
```


Figure 9.13: The first Bitcoin Block (Genesis Block) [https://blockexplorer.com/rawblock/, 2013-06-04]

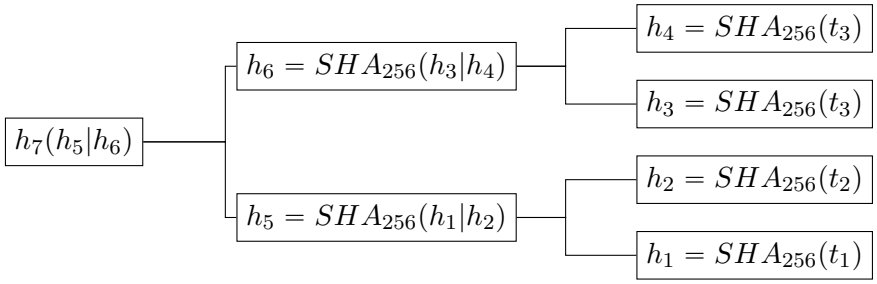


Figure 9.14: Example Merkle Tree

	Stack	Script
1.	λ	`<sig> <pubKey> OP_CHECKSIG`
2.	`<sig> <pubKey>`	`OP_CHECKSIG`
3.	true	λ

Figure 9.15: Script Execution for Transfer to IP

1. to 2.) and execution of the operation `OP_CHECKSIG` on the stack (Step 2. to 3.). If the stack is rendered into the value *true*, the Input is valid claim to this Output.

Transfer to Bitcoin Address

A Bitcoin-Address is the hash-value of public key, also called “fingerprint”. Verification of the transfer is done by comparing the fingerprint of a bc-address included in a transaction output-block with a public key given in the spending transaction block. Afterwards the owner of the public key proves ownership by signing with the related private key.

Input to Verification:

scriptPubKey: `OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`

scriptSig: `<sig> <pubKey>`

`<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`

subsectionSecurity of Bitcoin

Nakamoto calculates the probability of a successful attack, given single probabilities for honest, malicious nodes to find the next block. He provides an analysis that shows how it becomes harder for an attacker to manipulate the chain, the longer back he has to reach.

	Stack
1.	λ
2.	`<sig> <pubKey>`
3.	`<sig> <pubKey> <pubKey>`
4.	`<sig> <pubKey> <pubHashA>`
5.	`<sig> <pubKey> <pubHashA> <pubKeyHash>`
6.	`<sig> <pubKey>`
7.	`true`

Figure 9.16: Script Execution for Transfer to Bitcoin Address

Duplicate Transaction Vulnerability

It is possible, albeit highly difficult, to produce a transaction whose hash identifier equals a previous transaction identifier. The vulnerability has been registered under CVE-2012-1909 and lead to Bitcoin Improvement Proposal 30[9]. The solution was to change the acceptance rules for transaction in a way that transactions which had identifiers equal to previous transactions could not be included in a block unless the previous transaction was fully spend.

Double-Spending

Spending a single coin twice is actually not difficult, in a fast-payment scenario. This means, if you are with a merchant that is not waiting for a transaction to be confirmed in a block, for example if you are buying low value goods, e. g., food, you may have a second transaction prepared for the same coin, that is also not yet included in the blockchain, but will be, instead of the first one. Until the first transaction is rejected (and the merchant recognises this) you are long gone, having paid nothing. [**karame2012two**]

This attack, called *Race Attack* (Figure 9.17) even comes in two extensions that allow to double spend money even after confirmation. (See Finney and Vector76 attack.)

9.4.11 Privacy of Bitcoin

Bitcoin strictly speaking is not anoymous, but pseudonymous. Nakamoto, in his introduction paper [**nakamoto2008bitcoin**] states that a transaction is never linked to the identity of a user.

Transfer to IP Identity: PubKey (random)

Transfer to Bitcoin Addr. Identity: keyHash

[9] `https://github.com/bitcoin/bips/blob/master/bip-0030.mediawiki`

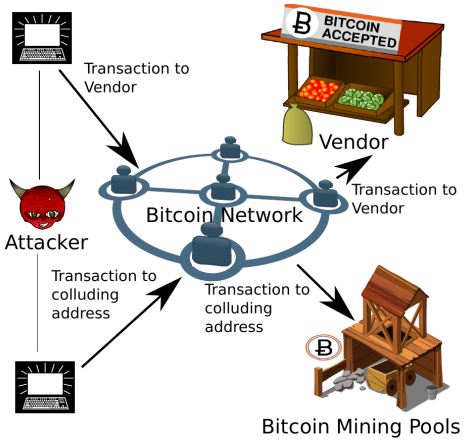


Figure 9.17: Race Attack for Bitcoin Double Spending [[**karame2012two**]]

- Pseudonymous — identified accounts
- money traces — transactions are linked
- → Zerocoin [**miers2013zerocoin**]

9.5 Conclusion

- No practical solution without ledger
- Traceability of transactions
- Proof-of-Work: “security by vote”
- Always on-participation

Exercises

Ex. 7 — Implement a Proof-of-Work generator and a verifier for a generic input string. (Use any hash algorithm that fits.)

Ex. 8 — Explain the objectives of the Proof-of-Work scheme.

Ex. 9 — Show that Proof-of-Work establishes asymmetric complexity at sender and recipient.

Ex. 10 — Sketch a network of example transactions similar to Figure 9.7 that distributes 70 Bitcoin among three participants. The first participant generates from the coinbase and has to pay 50 BC to the second and 20 BC to the third participant. The second participants pays 40 BC to the third participant. Finally

the third participant pays 10 BC and 30 BC to the second and first participant respectively.

Ex. 11 — What are possible and necessary attributes of electronic money?

Ex. 12 — Discuss the feasibility of an attack on the Blockchain.

Ex. 13 — Implement a prototype that evaluates a new block that is to be added to the Blockchain.

Ex. 14 — Discuss the effects of a non-loop-free stack language.

Ex. 15 — Discuss the trust-model of BitCoin using Audun Josang's trust models. Whom is trusted for what?

10

Web-Security

Contents

Lernziele:

- Secure Web-Communication
- WebID
- Web-Session
  - Session-Cookies
- Server-side Vulnerabilities
  - XSS/CSRF/...
- Client-side Vulnerabilities
- OWASP

10.1 OAuth

OAuth is an approach to authorise third parties to access defined rights to resources of an owner on a server. A very often used example is that of a photo printing service (the client), at which a user (the owner) using a browser orders a few prints of pictures (the resources) stored on a remote database (the server). Before OAuth, it was common, that the user had to provide any service to access his resources with his personal credentials to let the services client act on his behalf towards the server. As most often passwords are used for authentication, the service gained full access to the owner's resources for the whole validity period of the credential.

The main objective of OAuth is to allow for authorisation that is both restricted in its validity time, fine granular resource and rights selection. Disclosure of the password obviously is not sufficient.

In order to facilitate this, the owner, who is assumed to browse the clients webpage, has to authenticate himself using his server-password. The client then is provided with a token to authenticate itself for access to the resources towards

the server. The protocol uses HTTP Redirection to guide the owner from the clients web-page to the authentication pages of his (trusted) server and back to the client. The client explicitly has to state the requested access rights towards the server and the owner explicitly grants or denies these rights. In other words: the owner communicates directly with his server and not through the client.

resource owner (User) resource owner, the user that owns a web-resource and wishes to grant (temporary) access to that resource to the

client (Consumer) client, who might be a secondary webservice, a mobile app or similar, that should do something with a users resource hosted at a

server (Service Provider) server, that hosts and controls the resource owner's data.

(Terms in parentheses are used in the original spec, other terminology is from the RFC.)

client credentials

temporary credentials

token credentials

OAuth 1.0 RFC 5849 [**rfc5849**]Session Fixation Attack

OAuth 2.0 RFC 6749: OAuth 2.0, RFC 6750 OAuth Bearer Tokens

- als OpenID Connect auch zur Authentifizierung
- Facebook (seit 2012), Microsoft, Google,. . . experimental

OAuth provides a protocol to grant an application limited access for a limited time to a web-resource.

Sources State 2011:

OAuth 1.0 Session Fixation `http://oauth.net/advisories/2009-1/`

OAuth 2.0 **Facebook** `http://developers.facebook.com/docs/authentication/?_fb_noscript=1`

Google `http://googlecode.blogspot.com/2011/03/making-auth-easier-o.html`

Microsoft `http://windowsteamblog.com/windows_live/b/developer/archive/2011/05/04/announcing-support-for-oauth-2-0.aspx`

10.2 OAuth Phases

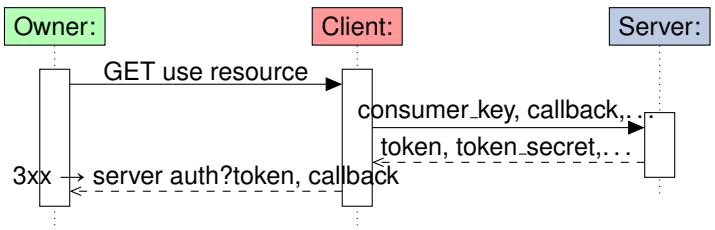


Figure 10.1: OAuth v1 Temporary Credential Request

Preliminaries:

0. (Client Credentials) *Client Credentials* are not part of the protocol, and are obtained beforehand to authenticate the client-software/instance.

Server Communication Endpoints:

1. Temporary Credential Request The first step in the protocol is for the client to request a *temporary credential*. These are used for Resource Owner Authorisation and afterwards by the client to claim a token from the server.
2. Resource Owner Authorisation The resource owner authorises the client by authenticating itself in a request to the server, including the clients temporary credential.
3. Token Request Using an previously authorised temporary credential, the client requests a token from the server for authentication of further requests.

10.2.1 OAuth Temporary Credential Request

As the owner requests some action from the client, that requires access to resources on the server, the clients sends a request to the server, requesting temporary credentials (in form of a token). The client then replies to the owner with a redirection to the authorisation endpoint of the server containing the token (not the token_secret). (Figure 10.1)

10.2.2 OAuth v1: Resource Owner Authorisation

The owner follows the clients redirection to the servers authorisation endpoint, using the token. There the owner authenticates himself to the server and explicitely authorises the client to access the given resource. The server then redirects the user to the client-callback previously set, providing a token_verifier. The owner calls (gets redirected) to the clients callback and thus issues the token_verifier to the client. (Figure 10.2)

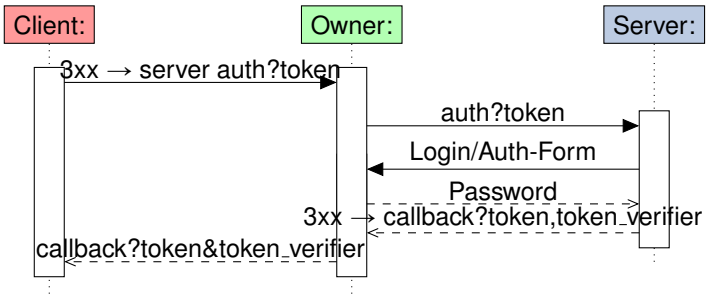


Figure 10.2: OAuth v1 Resource Owner Authorisation

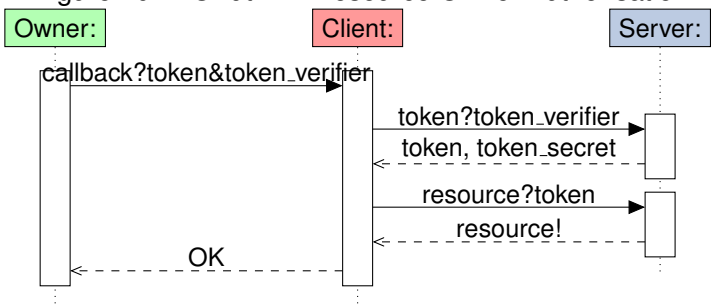


Figure 10.3: OAuth v1: Token Request

10.2.3 OAuth v1 Token Request

After the client received a token_verifier from the owner it requests an access token from the server, proving authorisation with the token_verifier. The server grants a new access token and the client can then request/use the resource. At the end it is nice to inform the owner of the result. (Figure 10.3)

- Attacker uses (honest) client to get temp. credential
- Attacker does not follow authorisation redirect
- Attacker tricks resource owner to click redirect
- Owner authorises honest client at server
- Attacker uses saved temp. credential to request token
- Attacker uses token to access resource

(source: http://oauth.net/advisories/2009-1/)

Why update?

- OAuth 1.0 too complex

- Scalability issues
- Incompatible to existing Auth. Schemes

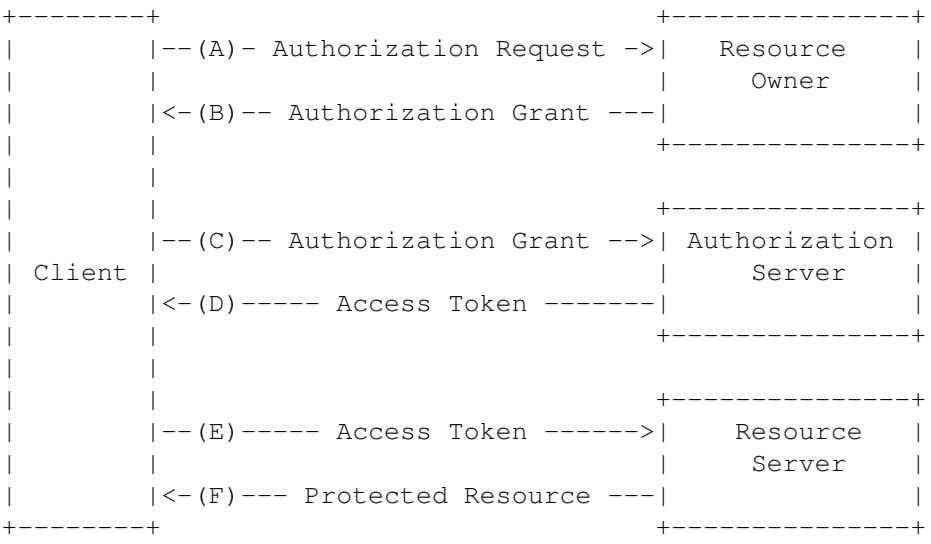
(source http://hueniverse.com/2010/05/introducing-oauth-2-0/)

State:

- Proposed Standard RFC 6749 (see https://datatracker.ietf.org/doc/html/rfc6749)
- Introduces an Authorization Server

- Role Separation: *Authorization Server*
- 4 Different Protocol Flows:
  - Authorization Code (OAuth 1)
  - Implicit (no client authentication)
  - Owner Password CredentialsCredentials are directly handled by the client. This gives the client full access to the resource server.
  - Client CredentialsClient uses own access to access resources. No owner authorisation required.
- Bearer Tokens
- Short-Lived Tokens/Long-Lived authorizations

(source http://hueniverse.com/2010/05/introducing-oauth-2-0/)



(source: RFC 6749)

10.3 Web-ID

In this lecture I want to introduce WebID a, in my eyes, very intriguing technologically beautiful — yet pragmatic — solution to the problem of identities, authentication and control over one's identity. The fundamental idea is, that everything should be identified by an URI and that, by that URI, further information, credentials and more, can be found. The idea takes up wide-spread technology that every computer user nowadays knows how to use — the World Wide Web, and enhances it in a way that is backward compatible and still provides new functionality.

WebID is an authentication scheme based on URI, SSL, x509 certificates, and the semantic web. It has been proposed by Dan Brickley and Tim Berners-Lee in 2000 and is supporting distribution of services as well as individual control and well defined representation by agents. It further is notably well suited to be used in `rdf:foaf`.

10.3.1 WebID Credentials

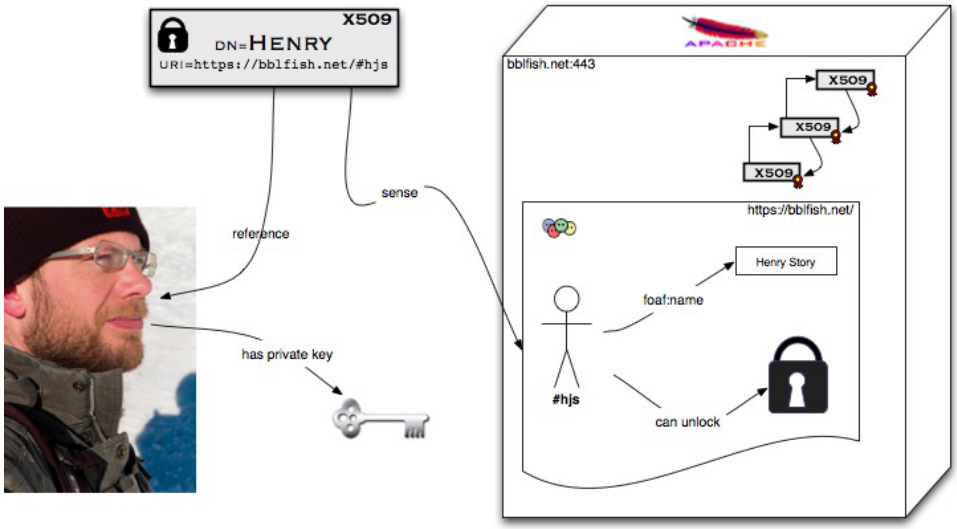
Henry has stored a public key in his FoaF-profile on his web server. He also has the matching private key stored locally accessible for the tools with which he accesses the web (which is mostly his browser). He further has the information stored on his server, that he has access to his profile.

10.3.2 WebID Authentication

WebID makes use of TLS described in Section**??** to authenticate both users (clients) and servers.

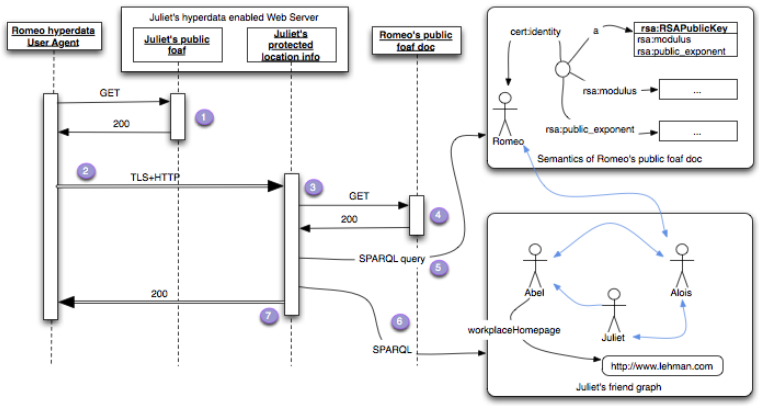
The authentication sequence in Figure 10.5 describes the process of Romeo accessing the homepage of Juliet. Juliet's server has a database that contains access control information in the form of a linked data graph. The steps are explained as follows:

1. In the first connection the user (Romeo) arrives on the home page of Juliet, after having received a note from her at a party. This page contains a login button or link.
2. Romeo clicks the login link, an https URL. Perhaps `https://juliet.example/login`
3. Juliet's server on receiving the HTTPS connection asks the client for his certificate. As a result a popup appears in Romeo's browser asking him to choose his WebId. (see the pictures of how different browsers present



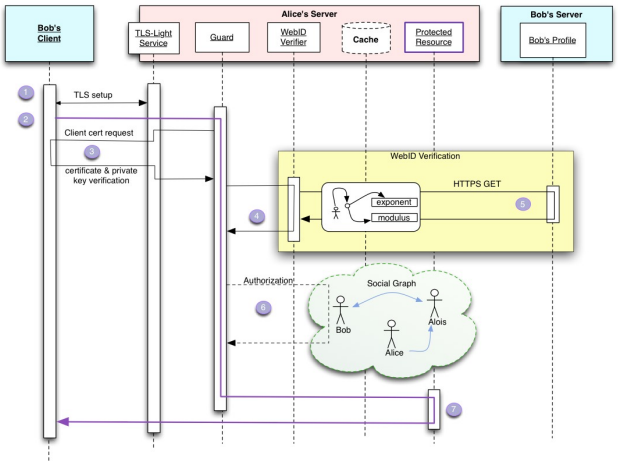
[http://www.w3.org/wiki/WebID]

Figure 10.4: WebID



[http://www.w3.org/wiki/Foaf%2Bssl]

Figure 10.5: WebID Authentication



[http://www.w3.org/wiki/Foaf%2Bssl]

Figure 10.6: WebID Authentication Sequence

this). Romeo selects one of them, and this corresponding X.509 certificate is sent back to the server, which verifies that Romeo's browser has the private key associated with the public key published in it, using the standard https protocol.

4. The certificate also contains a Web ID, which is a URL such as `http://romeo.example/#me` that was placed in the Subject Alternative Name field of the cert. Juliet's server finds this and does an HTTP GET on it.

5. If the graph of the returned document contains the relations that `http://romeo.example/#me` has the public key specified in the certificate, the server knows that the person at the other end of the connection is `http://romeo.example/#me` .

6. Juliet' server can then check in its database if `http://romeo.example/#me` is known by any of Juliet's friends, as specified in their foaf files published on their servers.

7. Juliet's server authorized Romeo to see her file.

We can, more specifically, take a look at the precise sequence of messages exchanged for authentication. The process is essentially a client certificate request that is happening within an SSL-secured session that should be commonly opened with every server now.

1. Bob's Client must open a TLS [RFC5246] connection with the server which authenticates itself using well known TLS mechanisms. This may be done as the first part of an HTTPS connection [HTTP-TLS].

2. Once the Transport Layer Security [TLS] has been set up, the application

protocol exchange can start. If the protocol is HTTP then the client can request an HTTP GET, PUT, POST, DELETE, ... action on a resource as detailed by [HTTP11]. The Guard can then intercept that request and by checking some access control rules determine if the client needs authentication. We will consider the case here where the client does need to be authenticated.

3. The Guard must requests the client to authenticate itself using public key cryptography by signing a token with its private key and have the Client send its Certificate. This has been carefully defined in the TLS protocol and can be summarised by the following steps:

   (a) The guard requests of the TLS agent that it make a Certificate Request to the client. The TLS layer does this. Because the WebID protocol does not rely on Certificate Authorities to verify the contents of the Certificate, the TLS Agent can ask for any Certificate from the Client. More details in Requesting the Client Certificate

   (b) The Client asks Bob to choose a certificate if the choice has not been automated. We will assume that Bob does choose a WebID Certificate and sends it to the client.

   (c) The TLS Agent must verify that the client is indeed in possession of the private key. What is important here is that the TLS Agent need not know the Issuer of the Certificate, or need not have any trust relation with the Issuer. Indeed if the TLS Layer could verify the signature of the Issuer and trusted the statements it signed, then step 4 and 5 would not be needed - other than perhaps as a way to verify that the key was still valid.

   (d) The WebID Certificate is then passed on to the Guard with the proviso that the WebIDs still needs to be verified.

4. The Guard then must ask the Verification Agent to verify that the WebIDs do identify the agent who knows the given public key.

5. The WebID is verified by looking up the definition of the URL at its canonical location. This can be done by dereferencing it. The Verification Agent must extract the public key and all the URI entries contained in the Subject Alternative Name extension of the WebID Certificate. A WebID Certificate may contain multiple URI entries which are considered claimed WebIDs at this point, since they have not been verified. The Verification Agent may verify as many or as few WebIDs it has time for. It may do it in parallel and asynchronously. However that is done, a claimed WebIDs can only be considered verified if the following steps have been accomplished successfully:

   (a) If the WebID Verifier does not have an up to date version of the WebID profile in the cache, then it must dereference the WebID using the canonical method for dereferencing a URL of that scheme. For an https://... WebID this would be done using the [HTTP-TLS] protocol.

   (b) The returned representation is then transformed into an RDF graph as specified in Processing the WebID Profile

   (c) That graph is then queried as explained in Querying the Graph. If the query succeeds, then that WebID is verified.

6. With the set of verified WebIDs the Guard can then check its access control rules using information from the web and other information available to it, to verify if the referent of the WebID is indeed allowed access to the protected resource. The exact nature of those Access Control Rules is left for another specification. Suffice it to say that it can be something as simple as a lookup in a table.

7. If access is granted, then the guard can pass on the request to the protected resource, which can then interact unimpeded with the client.

If the Client does not send a certificate, because either it does not have one or it does not wish to send one, other authentication procedures can be pursued at the application layer with protocols such as OpenID, OAuth, BrowserID, etc...

Move to fitting lecture

10.3.3 X.509 Certificates

Bind Name to public key

X509 `http://tools.ietf.org/html/rfc5280`

Formats: PEM, PKCS#7, PKCS#12

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 207481227 (0xc5de98b)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=DE, O=Universitaet Siegen, OU=Zentrum fuer Informations- und Medientechnologie, CN=Uni-Siegen CA - G02
        Validity
            Not Before: May 29 08:40:27 2008 GMT
            Not After : May 28 08:40:27 2013 GMT
        Subject: C=DE, O=Universitaet Siegen, OU=ZIMT, CN=xims.uni-siegen.de
        Subject Public Key Info:
                                        ⋮

        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c4:c7:af:46:87:7b:90:89:76:bc:6b:45:02:52:
                    2f:8d:54:da:68:c4:49:2b:4b:57:34:e9:c8:2f:4d:
                    bc:b5:28:25:66:1c:e8:26:db:b6:7a:88:b4:4f:ac:
                    2e:f5:a5:bd:92:93:51:09:f2:7e:96:b9:76:de:d5:
                    a3:9b:e2:fb:81:46:a9:d9:3b:ac:51:40:1f:68:6a:
                    b0:36:66:32:92:1b:14:74:08:77:c4:90:4a:54:19:
                    63:57:fa:29:70:2f:a6:c0:6b:36:c6:00:eb:85:ea:
                    90:c1:a1:50:aa:33:2b:db:e4:96:26:38:c1:e8:90:
                    82:45:ea:bc:13:a4:21:3d:05:b3:be:79:8e:bb:c3:
                    5b:51:96:c3:95:61:9f:b8:9f:ea:16:41:9e:c4:d6:
                    b4:1e:43:eb:e9:ff:cc:24:88:e1:44:64:af:b0:90:
                    9b:5f:77:1b:06:59:5d:0d:9a:0d:f5:e2:a4:7b:9b:
                    b1:42:58:c9:af:a0:ee:d6:e8:56:e6:48:97:05:dd:
                    80:97:40:08:cb:5e:7d:f1:ae:d2:05:c8:a3:67:1d:
                    43:ba:d8:3e:af:aa:ed:cf:4f:11:59:3b:b4:c2:3a:
                    dc:9a:6c:3e:1b:b6:c1:cd:d6:6d:bf:2c:cd:fc:b9:
                    ea:cb:b9:ff:31:68:32:58:18:23:0e:a6:8f:6a:92:
                    72:e7
                Exponent: 65537 (0x10001)
        X509v3 extensions:
                                        ⋮

        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Key Identifier:
                D3:9D:F5:70:C7:E0:14:00:3A:C7:2F:2F:4E:01:AB:53:DA:1F:C0:77
            X509v3 Authority Key Identifier:
                keyid:FF:74:C2:69:3A:F0:84:9F:9C:02:93:CD:9F:9E:F7:DD:FF:01:C5:65

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://cdp1.pca.dfn.de/uni-siegen-ca/pub/crl/g_cacrl.crl

                Full Name:
                  URI:http://cdp2.pca.dfn.de/uni-siegen-ca/pub/crl/g_cacrl.crl

            Authority Information Access:
                CA Issuers - URI:http://cdp1.pca.dfn.de/uni-siegen-ca/pub/cacert/g_cacert.crt
```

```
            CA Issuers - URI:http://cdp2.pca.dfn.de/uni-siegen-ca/pub/cacert/g_cacert.crt
                                        ⋮
    Signature Algorithm: sha1WithRSAEncryption
        4c:18:b0:04:2e:01:ae:67:d8:c4:79:cb:85:1b:a1:6d:ec:ff:
        ba:84:3c:e1:50:9d:95:91:b0:5e:ca:75:4c:6a:4f:69:0e:7e:
        c8:6f:eb:3e:2c:4e:e9:19:8b:35:9e:1f:19:0d:10:b4:88:a3:
        fb:8b:b4:f2:da:10:08:e0:83:4f:d8:15:90:5d:4a:b3:fd:10:
        2b:94:5b:79:61:e5:8e:d4:1d:4f:11:ac:c2:2a:44:bb:11:4e:
        2c:42:54:13:15:2a:a1:a5:bd:20:89:c4:83:8c:db:aa:66:28:
        5c:99:44:00:36:e1:1a:d9:a8:87:e8:a9:24:bc:56:39:63:0e:
        10:84:f2:03:7e:85:88:70:a1:2b:da:39:75:c5:b7:2f:3a:41:
        4f:b1:53:ba:c1:66:5c:0b:a0:5a:ff:0f:65:20:bd:b0:1f:2c:
        3d:42:ca:6a:f8:4c:73:af:20:93:98:9d:ca:a9:17:49:7a:9c:
        04:d8:5d:1e:2e:1b:36:85:f5:8f:83:a6:ab:49:ef:a5:2b:d0:
        7b:9e:80:a6:eb:87:1d:8f:16:79:d5:a2:4f:f1:e6:6e:4d:0c:
        ea:f1:a1:95:ec:db:dd:02:8e:41:14:9b:47:f6:6c:46:1a:f6:
        7b:85:9b:d6:80:0b:29:0e:54:b4:fb:e6:ab:2a:1b:09:64:aa:
        a4:44:3c:68
                                        ⋮
-----BEGIN CERTIFICATE-----
MIIFAjCCA+qgAwIBAgIEDF3pizANBgkqhkiG9w0BAQUFADCBhDELMAkGA1UEBhMC
REUxHDAaBgNVBAoTE1VuaXZlcnNpdGFldCBTaWVnZW4xOTA3BgNVBAsTMFplbnRy
dW0gZnVlciBJbmZvcm1hdGlvbnMtIHVuZCBNZWRpZW50ZWNobm9sb2dpZTEcMBoG
A1UEAxMTVW5pLVNpZWdlbiBDQSAtIEcwMjAeFw0wODA1MjkwODQwMjdaFw0xMzA1
MjgwODQwMjdaMFcxCzAJBgNVBAYTAkRFMRwwGgYDVQQKExNVbml2ZXJzaXRhZXQg
U2llZ2VuMQ0wCwYDVQQLEwRaSU1UMRswGQYDVQQDExJ4aW1zLnVuaS1zaWVnZW4u
ZGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDEx69Gh3uQiXa8a0UC
Ui+NVNpoxEkrS1c06cgvTby1KCVmHOgm27Z6iLRPrC71pb2Sk1EJ8n6WuXbe1aOb
4vuBRqnZO6xRQB9oarA2ZjKSGxR0CHfEkEpUGWNX+ilwL6bAazbGAOuF6pDBoVCq
Myvb5JYmOMHokIJF6rwTpCE9BbO+eY67w1tRlsOVYZ+4n+oWQZ7E1rQeQ+vp/8wk
iOFEZK+wkJtfdxsGWV0Nmg314qR7m7FCWMmvoO7W6FbmSJcF3YCXQAjLXn3xrtIF
yKNnHUO62D6vqu3PTxFZO7TCOtyabD4btsHN1m2/LM38uerLuf8xaDJYGCMOpo9q
knLnAgMBAAGjggGmMIIBojAJBgNVHRMEAjAAMAsGA1UdDwQEAwIE8DATBgNVHSUE
DDAKBggrBgEFBQcDATAdBgNVHQ4EFgQU0531cMfgFAA6xy8vTgGrU9ofwHcwHwYD
VR0jBBgwFoAU/3TCaTrwhJ+cApPNn5733f8BxWUwgYsGA1UdHwSBgzCBgDA+oDyg
OoY4aHR0cDovL2NkcDEucGNhLmRmbi5kZS91bmktc2llZ2VuLWNhL3B1Yi9jcmwv
Z19jYWNybC5jcmwwPqA8oDqGOGh0dHA6Ly9jZHAyLnBjYS5kZm4uZGUvdW5pLXNp
ZWdlbi1jYS9wdWIvY3JsL2dfY2FjcmwuY3JsMIGkBggrBgEFBQcBAQSBlzCBlDBI
BggrBgEFBQcwAoY8aHR0cDovL2NkcDEucGNhLmRmbi5kZS91bmktc2llZ2VuLWNh
L3B1Yi9jYWNlcnQvZ19jYWNlcnQuY3J0MEgGCCsGAQUFBzAChjxodHRwOi8vY2Rw
Mi5wY2EuZGZuLmRlL3VuaS1zaWVnZW4tY2EvcHViL2NhY2VydC9nX2NhY2VydC5j
cnQwDQYJKoZIhvcNAQEFBQADggEBAEwYsAQuAa5n2MR5y4UboW3s/7qEPOFQnZWR
sF7KdUxqT2kOfshv6z4sTukZizWeHxkNELSIo/uLtPLaEAjgg0/YFZBdSrP9ECuU
W3lh5Y7UHU8RrMIqRLsRTixCVBMVKqGlvSCJxIOM26pmKFyZRAA24RrZqIfoqSS8
VjIjDhCE8gN+hYhwoSvaOXXFty86QU+xU7rBZIwLoFr/D2UgvbAfLD1Cymr4THOv
IJOYncqpF0l6nATYXR4uGzaF9Y+DpqtJ76Ur0HuegKbrhx2PFnnVok/x5m5NDOrx
oZXs290CjkEUm0f2bEYa9nuFm9aACykOVLT75qsqGwlkqqREPGg=
-----END CERTIFICATE-----
                                        ⋮
```

10.3.4 Certificates in FoaF

```
<cert:key>
    <cert:RSAPublicKey>
    <cert:label>Lars Fischer</cert:label>
    <cert:modulus
        rdf:datatype="http://www.w3.org/2001/XMLSchema#hexBinary">
            BAAFB2E38A4E4FD49F9F0285D5929CA45EB1833607425E60CBB28AD
            31C61F146067989FBF3A47DB5B9D7E52C3AD337FC978093033A8E7B
            226AA6ACDC8FB6BBA7
    </cert:modulus>
    <cert:exponent
        rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
          65537
    </cert:exponent>
    </cert:RSAPublicKey>
```

```
</cert:key>
```

10.3.5 WebID Conclusion

- SSL based authentication
  - Browser has private key
  - any user action authenticated
- identifier: URI
- Webservices to write

10.4 Web Vulnerabilities

Every four years the Open Web Application Security Project (OWASP) publishes a list of the ten most dominant security flaws in web application implementations. Every four years this is somewhat of a sad reminder of how little the situation has changed — because one constantly finds “old fiends” at the top positions. Changes mostly happen when the Top-10-Project changes the classification of security risks.

Thus, make good aquaintance with the following, you might see each other again — and again — and again. And don’t ask “how could someone have made that mistake” ask yourself where you build this into code somewhere.

1. Injection. Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. Broken Authentication. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities temporarily or permanently.
3. Sensitive Data Exposure. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
4. XML External Entities (XXE). Many older or poorly configured XML processors evaluate external entity references within XML documents. Ex-

ternal entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

5. Broken Access Control. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6. Security Misconfiguration. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

7. Cross-Site Scripting (XSS). XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

8. Insecure Deserialization. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

9. Using Components with Known Vulnerabilities. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

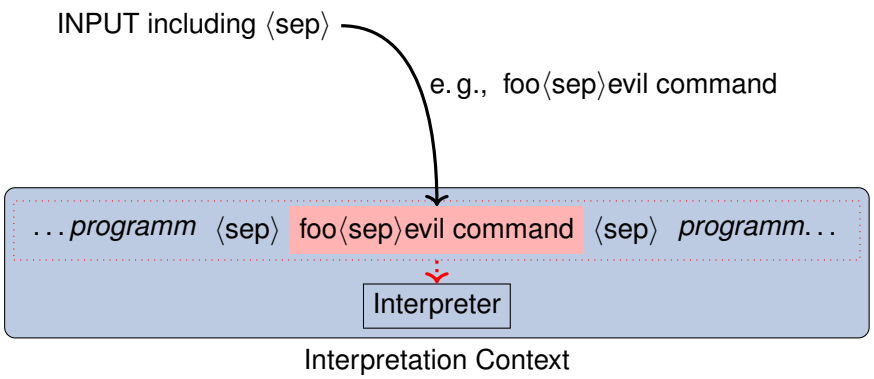
10. Insufficient Logging & Monitoring. Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

10.4.1 Server-side Vulnerabilities

Cross-Site-Scripting (XSS)

Nothing New about XSS

10.4.2 SQL-Injection



Interpretation Context

Code injections come in many flavours within many process environments. The most famous probably is the SQL injection.

Take a look at the example code.

```
@db.execute "INSERT␣INTO␣Games␣(secret,reward)"+
            "VALUES␣(\"#{secret}\",␣\"#{opts}\")"
```

Escape: "）

```
@db.execute("SELECT␣id␣FROM␣Games␣WHERE␣"+
            "secret=\"#{secret}\"␣and␣reward=\"#{opts}\""+
            "␣LIMIT␣1")
```

Escape: "

Auswahl von Mengen:

SELECT <row>[,row]* FROM <table> [WHERE <condition>]

Vereinigung von Mengen:

<Select_Statement1> UNION <Select_Statement2>

Conditions:

AND, OR, <row>=<val>

regexps, functions. . .

10.4.3 Client-side Vulnerabilities

Mobile Code; the general conception, that it is wise to execute someone else's code on your computer. That, generally speaking, is what happens when you open a webpage that has JavaScript embedded. And often

10.5 Attack Technology

In this section we will introduce some terminology from the colourful collection of attack tools.

In the following we extended the list found in [**Stallings**]


Advanced
Downloaders
Flooders
Keylogger
Auto-rooter
Persistent
Mobile
Threat
Watering
Backdoor
Spammer
Attack
Zombie
programs
Dropper
Trojan
Virus
Fork
Hole
Bomb
APT
Bot
Macro
horse
Logic
Spyware
Kit
Code
Worm
Rootkit
Adware
Exploits

Advanced Persistent Threat (APT) highly sophisticated threat action which maintains a foothold in the target system over a long time period. It is often ambigously used to also denote an attack group, i. e., the threat agent that executes, often repeated, attacks on a sophisticate technical level with substantial resources.

Adware

Air Gap Jumping for example through Near Sound Data Transfer (NSDT).

Attack Kit

Auto-rooter

Backdoor

Decompression Bomb Zip-Bomb

Downloaders i. e., Dropper

Drive-by-download

Dropper

Exploits

Downgrade Attack describes any attack technique that aims at enforcing the use of lower, i. e., insecure versions of algorithms, software, or infrastructure. SSL 2.0 was vulnerable to this kind of attack where a PitM was able to modify the list of accepted encryption algorithms in a way that the communication would not be encrypted (Null-Encryption).

Flooders (Stressor)

Fork Bomb

Keylogger

Logic Bomb

Macro Virus

Man-in-the-Middle (MitM) ??

Man-on-the-Side (MotS) [**maynard2020understanding**]

Mobile Code

Remote Access Tool (RAT) Posh name for backdoors

Rootkit a collection of tools designed to enable and persist access to a computer. Components may include, but are not limited to loaders, RAT, and information gathering tools.

Spammer programs

Spyware

Trojan horse

Virus malware that embedds itself within (benign) executable code, for example within an machine code, word-processor macros (Macro Virus).

Watering Hole

Worm

Blue Pill A tool, first implementation by Joanna Rutkowska, that inserts a minimalistic virtual machine between the OS and the hardware. Such an attack provides (undetectable) access to the lowest (most trusted) ring on a very fundamental level.

Dynamit-Phishing

Zombie, Bot

`https://www.heise.de/security/meldung/Malware-Emotet-Neuer-Dreh-sorgt`
`html?wt_mc=rss.red.security.security.atom.beitrag.beitrag`
Providing a URL in an email, leading to a malware download, in order to circumvent spam filters.

The following list is a reminder on more techniques:

- Credential Stuffing
- NOP Slide
- Buffer Overflow
- Integer Overflow
- Heap Spray
- Heap Feng Shui
- Shellcode (could be improved)
- MitM
- C2 Techniques
- Lateral Movement
- Dropper
- IDS Evasion
- NATPwn
- Clickjacking

- Server Side Request Forgery (SSRF): `https://owasp.org/www-community/attacks/Server_Side_Request_Forgery`
- Cross-Site Scripting (XSS):

  insert slides from 35c3 presentation

10.6 Secure SW-Dev

Only a secure program is a good program. This collection provides an entry do different *secure coding practices*.

> "any person can invent a security system so clever that he or she can't think of how to break it" (interpretation by Cory Doctorow in 2004) [1]

Solitaire

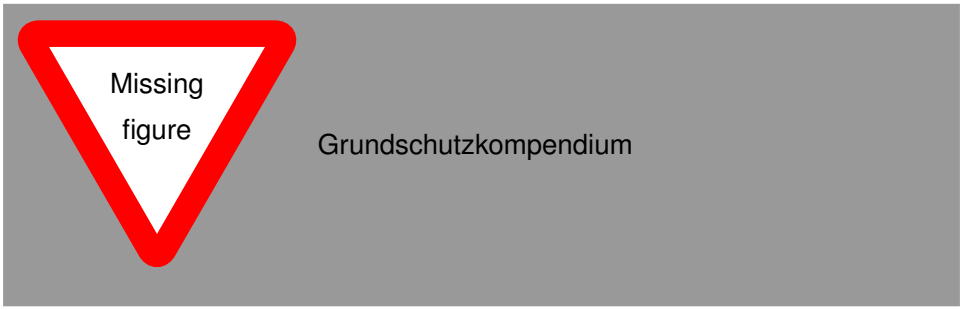
has been an encryption algorithm created by Bruce Schneier, that could be executed manually using an ordered deck of cards. It was published in the appendix of the book "Cryptonomicon" by Neal Stephenson in 1999. The storyline and motivation was that a deck of cards would be a perfectly insuspicious item to be carried around by secret agents. The ordering of the deck would would implement a specific key and could either be stored physically (in the deck) or memorized.

Interestingly Schneier's Law from 1998 should find a perfect example when Paul Crowley found vulnerabilities in *Solitaire* less than a year after its publication, still in 1999.

- OWASP SAMMSecurity Assurance Maturity Model
  - Security Practices for:
    - Governance
    - Design
    - Implementation
    - Verification
    - Operations
- MITRE: Systems Engineering Guide
- Common Criteria

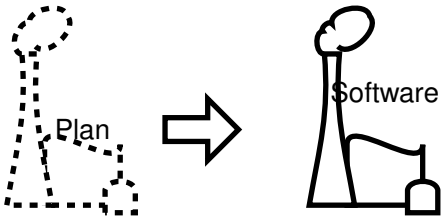
[1] The original quote read "Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break." and has been phrased in 1998 [`https://www.schneier.com/blog/archives/2011/04/schneiers_law.html`]

- ...
- BSI Grundschutzkompendium
- CON.8 Softwareentwicklung
  - Entwicklungsumgebung/-prozesse
  - Vertrauenswürdige Bibliotheken
  - SW-Development Chain
  - Sicherheitsschulung



10.6.1 Systemdesignprinzipien

At the bottom the process of creation can be boiled down to three phases: planning, building and setting the thing up. You might that this is only one step short of the full PDCA-cycle that you know from management because in this model we are not concerned with running and maintaining our thing.



Basic engineering phases:

- Concept development
- Engineering development
- Post-development

[MITRE: Systems Engineering Guide]

Design ist die Überführung des Ist-Zustandes in einen Soll-Zustand.

- Hohe Komplexität

Principle	Explanation
Open design	Kerckhoff 2nd PrincipleAssume the attackers have the sources and the specs. Build your processes in a way that provides security that is not only based on obscurity of processes but defined secrets of determined strength.
Fail-safe defaults	Fail closed; no single point of failure.
Least privilege	Need-2-Know Principle: No more privileges than what is needed.
Economy of mechanism	Keep it simple, stupid.
Separation of privileges	Don't permit an operation based on a single condition.
Total mediation	Check everything, every time.
Least common mechanism	Beware of shared resources.
Psychological acceptability	Will they use it?

Table 10.1: Saltzer/Schroeder Security Design Principles [source: `http://msdn.microsoft.com/en-us/magazine/cc163519.aspx`, 2013-09-26]

- Abhängigkeiten
  - Funktionen
  - Variablen
  - Module
  - Bibliotheken
  - Kommunikationspartner
  - Physische Umgebung
  - Nutzer / Bedienpersonal
- Maintenance / Updates ?

See Table 10.1.

BSI Empfehlungen Seit 2020 hat das Bundesamt für Sicherheit in der Informationstechnik (BSI) Empfehlungen für den Grundschutz in der Software-Entwicklung als Schutzmodul CON.8 aufgenommen. [**BSI2020Grundschutzkompendium**]

Diese Beziehen sich aber auf den Aufbau und den Betrieb von Software-Entwicklung. Empfehlungen für gute Coding-Praxis sind nicht enthalten.

1. Validate Input
2. Heed compiler warnings
3. Architect and design for security policies
4. Keep it simple
5. Default deny
6. Adhere to the principle of least privilege
7. Sanitize data sent to other systems
8. Practise defence in depth
9. Use effective quality assurance technique
10. Adopt a secure coding standard

Bonus:

- Define security requirements
- Model threats

OWASP Secure Coding Practices

Quelle: `https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices`,

OWASP provides a exhaustive checklist for secure coding.

1. Input Validation
2. Output Encoding
3. Authentication and Password Management (includes secure handling of credentials by external services/scripts)
4. Session Management
5. Access Control
6. Cryptographic Practices
7. Error Handling and Logging
8. Data Protection
9. Communication Security
10. System Configuration
11. Database Security

12. File Management
13. Memory Management
14. General Coding Practices

Quelle: OWASP Coding Guidelines [**OWASP2010SecureCodingPractices**]

What is a secure coding standard?

from: `https://security.berkeley.edu/secure-coding-practice-guidelines`

Input Validation

It sounds like a no-brainer, but often enough is found lacking in code: If you transfer data from one context to the next, make shure it contains what is expected and does not contain anything that might have unintended effects at the destination.

The default way to handle input validation should be, as always, be guided by the Default-Deny principle. The programmer must specify the set of allowed symbols (a whitelist) and should not express the forbidden exceptions (blacklist).

Example (whitelisting in Ruby)

```
unless (/^[a-zA-Z0-9., ]$*/ =~ input) then
    handle_fail
else
    use_input
end
```

Encode HTML Output in Ruby

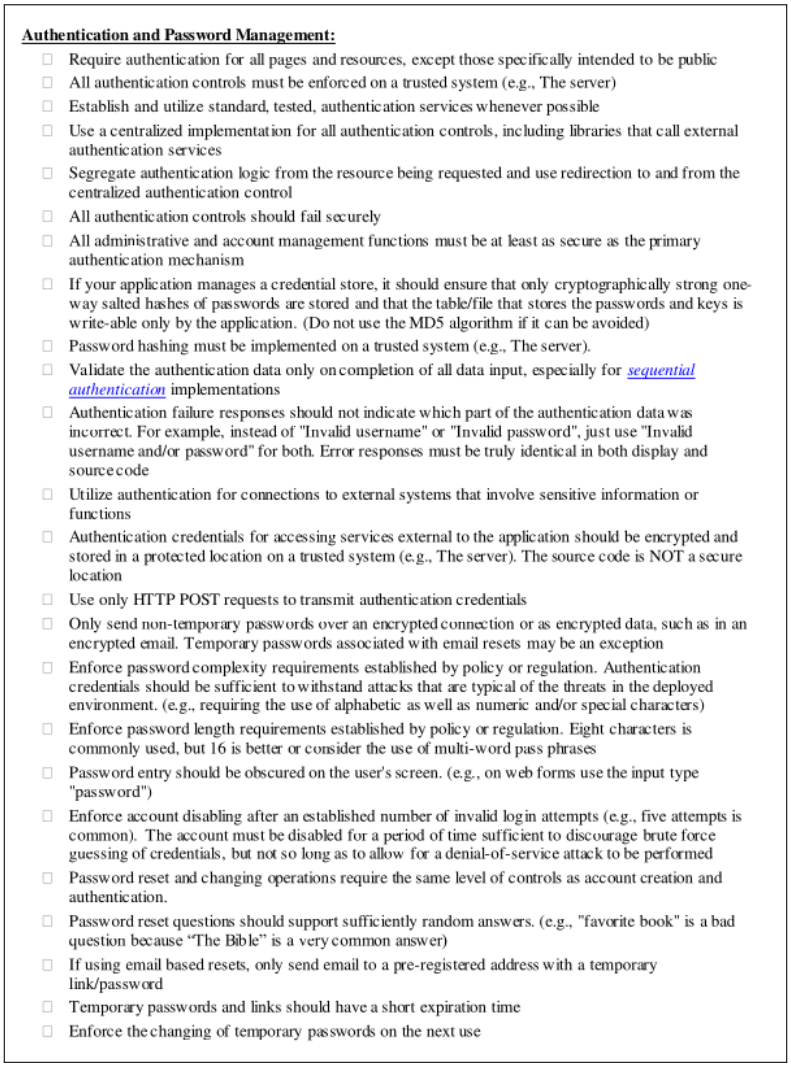
```
output = 'bla<script>alert()</script>'

{">" => "&gt;", "<" => "&lt;"}.each {
                |c, e| output.gsub!(c, e)}

output
 => "bla&lt;script&gt;alert()&lt;/script&gt;"
```

HTML-Entities:

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
```



Authentication and Password Management:

- ☐ Require authentication for all pages and resources, except those specifically intended to be public
- ☐ All authentication controls must be enforced on a trusted system (e.g., The server)
- ☐ Establish and utilize standard, tested, authentication services whenever possible
- ☐ Use a centralized implementation for all authentication controls, including libraries that call external authentication services
- ☐ Segregate authentication logic from the resource being requested and use redirection to and from the centralized authentication control
- ☐ All authentication controls should fail securely
- ☐ All administrative and account management functions must be at least as secure as the primary authentication mechanism
- ☐ If your application manages a credential store, it should ensure that only cryptographically strong one-way salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. (Do not use the MD5 algorithm if it can be avoided)
- ☐ Password hashing must be implemented on a trusted system (e.g., The server).
- ☐ Validate the authentication data only on completion of all data input, especially for *sequential authentication* implementations
- ☐ Authentication failure responses should not indicate which part of the authentication data was incorrect. For example, instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both. Error responses must be truly identical in both display and source code
- ☐ Utilize authentication for connections to external systems that involve sensitive information or functions
- ☐ Authentication credentials for accessing services external to the application should be encrypted and stored in a protected location on a trusted system (e.g., The server). The source code is NOT a secure location
- ☐ Use only HTTP POST requests to transmit authentication credentials
- ☐ Only send non-temporary passwords over an encrypted connection or as encrypted data, such as in an encrypted email. Temporary passwords associated with email resets may be an exception
- ☐ Enforce password complexity requirements established by policy or regulation. Authentication credentials should be sufficient to withstand attacks that are typical of the threats in the deployed environment. (e.g., requiring the use of alphabetic as well as numeric and/or special characters)
- ☐ Enforce password length requirements established by policy or regulation. Eight characters is commonly used, but 16 is better or consider the use of multi-word pass phrases
- ☐ Password entry should be obscured on the user's screen. (e.g., on web forms use the input type "password")
- ☐ Enforce account disabling after an established number of invalid login attempts (e.g., five attempts is common). The account must be disabled for a period of time sufficient to discourage brute force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed
- ☐ Password reset and changing operations require the same level of controls as account creation and authentication.
- ☐ Password reset questions should support sufficiently random answers. (e.g., "favorite book" is a bad question because “The Bible” is a very common answer)
- ☐ If using email based resets, only send email to a pre-registered address with a temporary link/password
- ☐ Temporary passwords and links should have a short expiration time
- ☐ Enforce the changing of temporary passwords on the next use

Figure 10.7: Example page from the OWASP Secure Codinc Practise

11

Privacy

Contents

Lernziele:

- Identität, Pseudonymität, Anonymität
- Quasi-Identifier
- De-Anonymisierung
- Anonymisierungsdienste
- Privatheit bewerten
  - Anonymity Set Size
  - Bewertung von Datenarten

In this lecture we will discuss privacy in communication technology. Starting from a definition and motivations for privacy, we try to find the meaning of privacy by discussing measures. After all, if you can quantify it, you should have an idea what the concept is. We also strive out to the technologies that make privacy happen (mostly communication mix networks).

Privacy is not a precisely defined term. Most people seem to associate something about hiding information about themselves with it. Although most techniques we are about to learn deal exactly with this problem, I also like to stress, that privacy is also, or much more, about controlling the way others are able to perceive us. It is very much difficult to completely hide oneself and still participate in social activities, but then it normally is not desirable to even be able to completely vanish. Thus privacy also is more about which information is disclosed about us.

as we will see and loathe below

11.1 Definitions

For a start it would be nice to know what all this privacy is about. Sadly, the definition of privacy turned out to be controversial.

Privacy is foremost, and started as, a problem of society.

11.1.1 Privacy Paradigms

The semantics of what actually is understood as privacy seemingly is complicated. Different paradigms have been followed over time. And while semantics seem like splitting hairs — everybody understands what *privacy* is — this has a substantial impact on objectives and thus technologies underneath. Like most innovation steps in the field, this is driven by researchers showing the shortcommings of former paradigms by successfull development of attacks.

- Privacy as Confidentiality, “if only you keep your data secret, you have sufficient privacy.” Has become obsolete when it was shown, that meta-data is just as revealing as the original data. Thus it is time for the next paradigm:
- Privacy as Control, “if only I can control who is seeing which information on me, I have privacy”. Has been the dominant paradigm that led to the development of anonymising network, like Tor or Mixmaster. Nonetheless, it turned out that controlling meta-data is not sufficient, if your behaviour is revealing most of it.
- Privacy as Practise, “it is not sufficient to deploy technological gadget, people have to practise privacy-aware behaviour.”

11.1.2 Privacy — Socially

- “The right to be left alone’ [Warren/Brandeis 1890] Probably is the first mention of the right of individuals to privacy. Previous guidelines where, for example medieval etiquette only provided rules for what one was responsible to *not to show*. There also where certain rules on when someone should look aside, but these guidelines where only describing curtsies and not general rules or personal rights.
- Informational Self-Determination (Informationelle Selbsbestimmung)[3mm]

  > “die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen” — BVerfGE 65, 1 - Volkszählung (1983)

  Much later, in 1983, the Bundesverfassungsgericht, the highest court in Germany, derive a general right for privacy, the *Informationelle Selbstbestimmung* directly from the federal constitution. This right for „informational self-determination” directs that every individual generally is able to decide about the usage of data on itself.

  We are no lawyers, so just call it constitution, although it formally is not.

  > “Everyone has the right to respect for his private and family life, his home and his correspondence.” — ECHR Art. 8 (1)

ECHR: European Convention on Human Rights. Auch wenn das deutsche Grundrecht auf I.S. im eigenen Land noch keinen expliziten Gesetzestext erhalten hat, ist Privatssphäre über die Europäische Menschenrechtskonvention auch in Deutschland rechtlich angelegt. Allerdings bezieht sich der Artikel in Absatz 2: “There shall be no interference by a public authority with the exercise of this right except. . . ” direct auf öffentliche Authoritäten und schliesst damit vermutlich private Firmen aus, diese sind aber, zumindest in OSN die Instanzen, welche über große Datenbestände verfügen können.

another (practise-based) definition

“freedom from unreasonable constraints on the construction of one’s own identity” [Richard Clarke: The digital persona and its application to data surveillance]

11.1.3 Privacy — Technically

The term privacy itself is very much undefined from a pure technological point of view. It stands to reason that it might not be sensible to even try to have a technical definition for this term. But we can define sub-problems of the terminology.

11.1.4 Quasi-Identifier

Identity An ambigiously used term that is, in social context, often used to refer to a distinct personality. More common it refers to a distinct continuity of existence, e. g., a human being. But most interpretations are based on the philosophical and mathematical definitions, which denote by identity a concept of sameness, by which two things are recognisable as separate or same things. Most fundamental is the formal identity function which defines which things are considered equal.

Identifier An attribute that is chosen for its unique value it has for each entity. Usually an identifier has no other purpose or semantics as providing this unique value. But, depending on the specific handling identifiers might also carry additional information, e. g., order of membership entry or access rights[1].

Quasi-Identifier A set of attributes whose values are unique for a subset or all individuals. Quasi-identifiers are often not *intended* to identify people. For example a set of columns in a relational database with a combination of values that is unique or almost unique for individuals within a set.

[1] We are obviously playing with the Double-Zero identifiers from the famous novels of Ian Flemming that indicated the “Licence to Kill”.

11.1.5 Anonymity and Unlinkability

- Privacy — undefined
- Anonymity

  > "Anonymity of a subject means that the subject is not identifiable within a set of subjects, the anonymity set." — [**Pfitzmann2010**]

- Unlinkability

  > "Unlinkability of two or more IOI from an attacker's perspective means that within the system [...], the attacker cannot sufficiently distinguish whether these IOIs are related or not." — ibid.

"Linkability" and "Unlinkability" are artificial terms to denote the hardness or simplicity to find the relation between items of interest.

These definitions a based on formal models namely the Mix-Anonymity Model in Section 11.3.2.

11.1.6 Unlinkability Problem

Especially the MIX anonymity model, but all the other models as well, resemble, more generally, assignment problems known from graph theory. Figure 11.1 shows an example assignment as *anonymity problem* which consists of a set of artefacts on the left side and a disjoint set on the right side. These anonymity problems resemble situations where starts and ends are to be connected by the attacker by inferring connections or transitions, for example the sender and receiver in the Mix-Anonymity Model in Section 11.3.2.

In [**Fisch2012MeasuringUnlinkabilityPrivacy**] I distinguished these types of problems from *unlinkability problems* where an attacker seeks to derive an equality relation within a single set of items of interest. An easy example is a set of messages and the question which messages have the same senders.

Anonymity Problem:

Linkability Problem:

Mapping Anonymity onto Unlinkability Problems

In practise unlinkability problems resemble situations in which we have even less information than in anonymity problems (we do not even know the names).

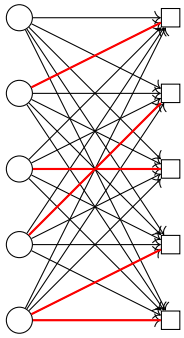


Figure 11.1: Anonymity Problem

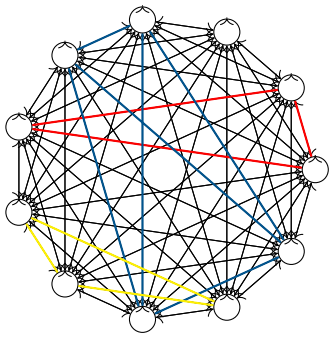


Figure 11.2: Unlinkability Problem

One of the advantage of our model as graph theoretic models is, that we can easily see that, in theory, unlinkability problems are much more general as we can model any anonymity problem as an unlinkability problem. In practise this phenomenon justifies the notion of quasi-identifiers (see Section 11.1.4) as it shows how the linkability of artefacts reduces the requirement for explicit identifiers. For attackers the question of the name often is of lesser importance to deriving the communication history of a subject.

Many known attacks are based on linkability, generally all attacks that, for example, use similarity measures to incrementally extend a hitting set. Examples are the clustering algorithms given in Section **??**.

In [**Fisch2012MeasuringUnlinkabilityPrivacy**] we gave the following proof by construction.

Assume that an attacker is unable to solve anonymity problems, but owns an oracle that perfectly solves unlinkability problems. The attacker can express every anonymity problem by constructing the disjoint union $M_U := M \cup^* U$, where M is a set of IOI and U is a set of IA. The oracle then solves the problem to determine the real partition from Π_{M_U}.

Additionally the attacker may even reduce the complexity of the unlinkability problem by using the hint-class "breach of unlinkability" from [**franz07:˙attac˙unlin**]. This class describes the situation where an unlinkability-attacker gets to know a set of elements that pairwise are in different equivalence classes. We denote the set of set partitions $\Pi_{M_U}(\mathcal{H}_U)$ of a set M as conditioned by the hint $\mathcal{H}_U$. The hint $\mathcal{H}_U$ describes, that no two elements in of the set U are in the same cluster. The hypotheses space then is defined in [**franz07:˙attac˙unlin**] by

$$\Pi_{M_U}(\mathcal{H}_U) := \{\pi \in \Pi_{M_U} : \forall\{m, m'\} \subseteq U \Rightarrow m \not\sim_\pi m'\}.$$

Where $m \not\sim_\pi m'$ denotes that m and m' are not in the same equivalence class with respect to partition π.

Knowing the subject identifiers U hint $\mathcal{H}_{|U|}$, which defines that the number of clusters is equal to $|U|$ can be applied. This hint reflects the common global anonymity scenario where all subject identifiers are known and each item of interest has to be related to exactly one subject. This can be modelled as hint "number of equivalence classes". Combining both hints, a restricted unlinkability hypotheses space of set partitions can be defined as

$$\Pi_{M_U}(\mathcal{H}_U, \mathcal{H}_{|U|}) = \{\pi \in \Pi_{M_U} : |\pi| = |U| \text{ and } \forall\{m, m'\} \subseteq U \Rightarrow m \not\sim_\pi m'\}. \tag{11.1}$$

Where $|\pi|$ denotes the number of clusters, i. e., equivalence classes, in a set partition π.

Even without these hints the oracle would produce the real partition. By re-identifying the IA in M_U the attacker thus is able to derive the mapping from IOI to IA and thus has solved his original anonymity problem.

Thus, we have shown that any anonymity problem can be mapped to an unlinkability problem. Obviously the other direction is not possible because, as the above construction shows, the set of hypotheses in anonymity is but a subset of the unlinkability hypotheses set.

11.1.7 Undetectability and Unobservability

By *undetectability* we mean that an attacker is not able to decide whether an item of interest, e. g., a communication event, exists or not. A communication event is *unobservable*, when, additionally, an attacker is not able to determine which entities communicate with each other, i. e., the involved communication partners are anonymous. [**Pfitzmann2010**]

11.1.8 Privacy Technologies

Privacy Techs classified in [**DanezisGuerses2010**]

- PETs
  - Communication Anonymity and Anonymisers
  - Identity Management Systems
  - Privacy Policy Languages and Privacy Negotiations
- Privacy Preserving Data Publishing and Mining
  - Tabular Data Analysis:
    - Privacy Preserving Data Publishing: forestalling inference by suppression or generalisation of data
    - Privacy Preserving Data Mining: randomisation, perturbation and cryptography to allow data mining on non-sensitive information
  - Differential Privacy: indifference of data with respect to containing data of a specific person
  - Social Network Analysis: global, local and injection attacks
  - Privacy Preserving Recommender Systems

11.2 Privacy Attacks

11.2.1 Fingerprinting/Tracking

How is a web page operator able to track you? The key to tracking is re-identification, i. e., recognition of different communication events as belonging to the same user.

- (Quasi-)identifier
- Stochastic inference

11.2.2 Inference attacks

Statistical attacks based on accumulation of much data.

11.3 Models

- Anonymity Set
- MIX-Communication Model
- PROB-Channel
- Unlikabilty-Anonymity

11.3.1 Anonymity Set

The *anonymity set* provides the most fundamental model of anonymity. It has been conceived by David Chaum in *The Dining Cryptographers Problem*[2] in which he defines the term *dining cryptographers network* (or DC-net for short).

Cardinality of Set of possible Senders

> An *anonymity set seen by a set of keys* is the set of vertices in a connected component of the graph formed from the original graph by removing the edges concerned.

(The Dining Cryptographers Problem [Chaum 1988])

Dining Cryptographers Network

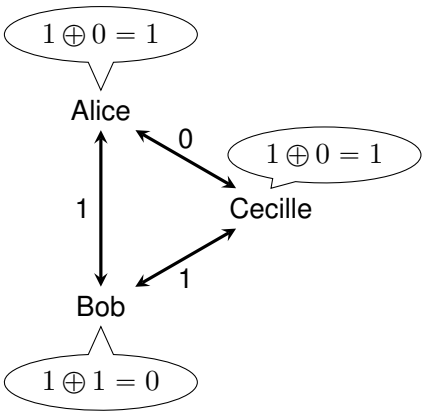
Three cryptographers have lunch at a restaurant. When they ask for the bill, the waiter announces that the bill has already been paid. Suspiciously they want to find out if one of them has paid the bill anonymously or some external entity. The protocol is quite simple, but relies on the well-behaviour of its participants.

First each pair of cryptographers secretly throws a coin, e. g., behind a menu. Then each cryptographer knows two boolean values, each with one of the other

[2] a play on the Dining Philosophers Problem by Dijkstra

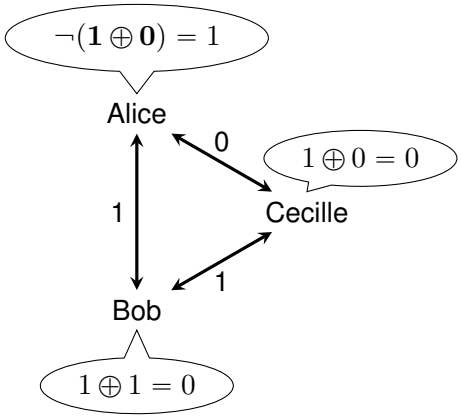
cryptographers. Each cryptographer then announces the result of XORing the two coin values known. The only difference is, that the cryptographer who secretly paid the bill is announcing the inverse of the calculated value.

Now every cryptographer calculate the XOR of the three public values. The result will be 0 (false) if no cryptographers announced the inverse value, i. e., because none of the three paid the bill. If exactly one of the cryptographers has announced the inverse value, because he paid the bill, the XOR of all three values will be 1 (true).



$1 \oplus 1 \oplus 0 = 0$ (someone else paid)

(a) Dining cryptographers network when an external entity paid the bill.



$0 \oplus 1 \oplus 0 = 0$ (Alice paid)

(b) Dining cryptographers network when Alice paid the bill.

11.3.2 MIX-Model

The MIX network describes the unlinkability of communication endpoints as the linkability of IoIs that pass through an unobservable area. This area can be

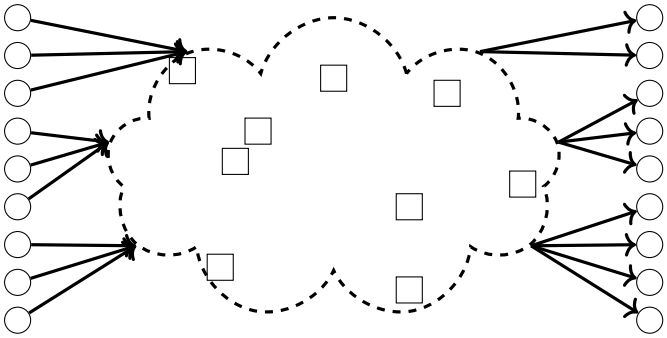


Figure 11.4: Mix-Anonymity Model [**Pfitzmann2010**]

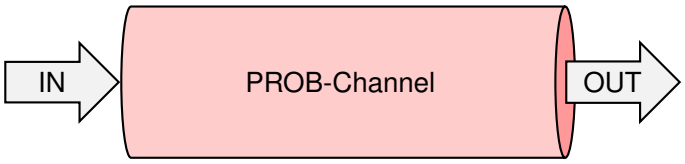
implemented by specific buffering routers, MIX-nodes, that obfuscate statistic correlation of incoming and outgoing communication events, e. g., data packets.

This is the fundamental concept to describe mix networks, e. g., The Onion Router (Tor).

MIX-Model[5mm]

11.3.3 PROB-Channel

The PROB-Channel provides a time-based correlation analysis model to correlate Item-of-Interests (IoIs) travelling through an unobservable channel. This is very similar to the MIX-model.



Residence Time δ

Density of Residence Time: $f(\delta)$: $\sum_{0}^{\infty} f(\delta)d\delta = 1$ Der PROB-Channel modeliert die Situation eines Beobachters der Endpunkte eines Anonymisierdienstes, der keinerlei Einblick in das Innenleben des Anonymisierdienstes hat. Die entscheidende Eigenschaft um eingehende und ausgehende Signale miteinander zu verketten ist die Verweildauer δ im PROB-Channel.

Measurement:

IN/OUT	in_0	in_1	$\dots$
out_0	$\delta_{0,0}$	$\delta_{0,1}$	$\dots$
out_1	$\delta_{nein1,0}$	$\delta_{1,1}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$

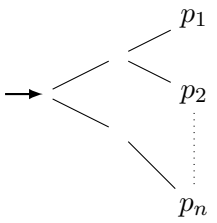


Figure 11.5: Potential Paths of a Message

Attacker's Solution: Minimum Assignment with Respect to expected RT Der Beobarchter kann nun eine Anzahl von Nachrichten, die sich zum selben Zeitpunkt im Channel befunden haben müssen versuchen zuzuordnen. Möglichkeiten bieten hier zum Beispiel maximale Assignments.

11.4 Privacy Measures

Under the topic of *privacy in communication* we address techniques to hide the contents of communication as well as information about connection endpoints and interaction context. Simplified the field is dealing with the questions of "who" is communicating "what", "when" and "where" and methods to control dissemination of related information. The term for data that describes the circumstances of a communication event is *metadata*.

Topics related to metadata in privacy are *sender/receiver anonymity* and *unlinkability* of communication events.

Research in privacy measures has been an iterative process. Researchers find situations that where known measures fall short to reflect the differences and then propose a new measure. This section is roughly reflecting this process, starting with the fundamental anonymity set size.

11.4.1 Sender/Receiver Anonymity

Based on the Mix-Anonymity Model we can derive different measures for anonymity in communication.

A communication consists of two or more communication endpoints and an item-of-interest (e. g., the message). Communication endpoints are commonly distinguished as sender or receiver and privacy objectives differ whether sender or receiver should be hidden from the partner or observers.

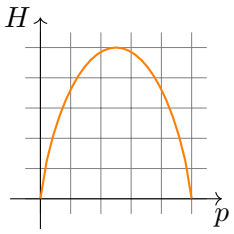


Figure 11.6: Entropy of two propabilities: $p, (1-p)$

11.4.2 Anonymity Set Size

The *size of an anonymity set* is a basic measure for anonymity. Anonymity sets, as given in Section 11.3.1consist of potential entities in a relation to an event, e. g., potential senders of a message, for which an attacker could not distinguish which individual within the set is the actual related entity.

The set size represents the fundamental idea of hiding in numbers, if the flock is large enough, the individual should not be found. This seems to be a valid assumption if only the entities within the anonymity set are indistinguishable.

11.4.3 Probability Anonymity Set

The size of the Anonymity Set may be a very descriptive term, but it falls short in situations where there is uncertainty whether an entity belongs to the set or where an attacker can derive different probabilities for entities within the set (i. e., distinguish entities within the set). The situation displayed in Figure 11.5 shows a simple example for the latter situation where, given uniform distribution at the forks, the probability of an item of interest being related to exactly the lowest path is more likely compared to the other paths.

Entropy of a probability distribution on the members of an anonymity set is used as a measure for privacy. (See Section **??** for the main section on Shannon Entropy.)

$$H(X) = \sum_{i=1}^{N} p_i \log_2 (1/p_i)$$

(Vgl. [**shannon48entropy-orig**], S. 1-12)

11.4.4 Degree of Anonymity

Shannon's Entropy is, as has been intended, sensible to the number of elements in the base set, i. e., if the size of the anonymity set is larger, so is the

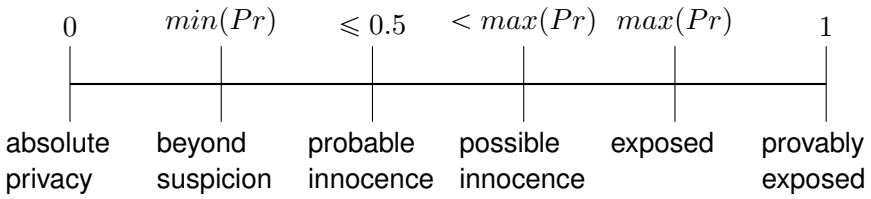


Figure 11.7: Scale of the Individual Anonymity Degree [**reiter1998:crowds:anonymityweb**]

entropy. This is not always wanted in anonymity measures where not an absolute minimum code length is requested, but a general quantification a solution. Thus in 2002 Claudia Diaz tackled this problem by introduction of the *Degree of Anonymity*.

Degree of Anonymity $D(P) = \frac{H(X)}{\log_2(N)}$ (Vgl. [**diaz02:˙towar**])

11.4.5 Combinatiorial Anonymity

There is a different line of research in privacy metrics that is not using bayesian stochastic but relies on combinatorial calculations to estimate the number of tries an attacker requires for a successful attack.

- Combinatorial Anonymity

  e.g.: How many samples are needed?

  Disclosure Attack

  - Distinct communication partner of Alice
  - potential Recipient/Message
  - Exclusion of Hypotheses by Observation

(Vgl. [**Agrawal2003**])

11.4.6 Individual Anonymity Degree

11.4.7 Unlinkability Measures

Unlinkability measures have been researched much later than anonymity measures. The first distinguished work on this problem by Sandra Steinbrecher und Stefan Köpsel adapted the Degree of Anonymity onto the set of all partitions of the set of items of interest. A partition of the set can be interpreted as the definition of an equivalence relation. Thus this Degree of Unlinkability provided a quantification of the certainty with which an attacker would choose a certain equivalence relation. [**steinbrecher03:˙model˙unlin**]

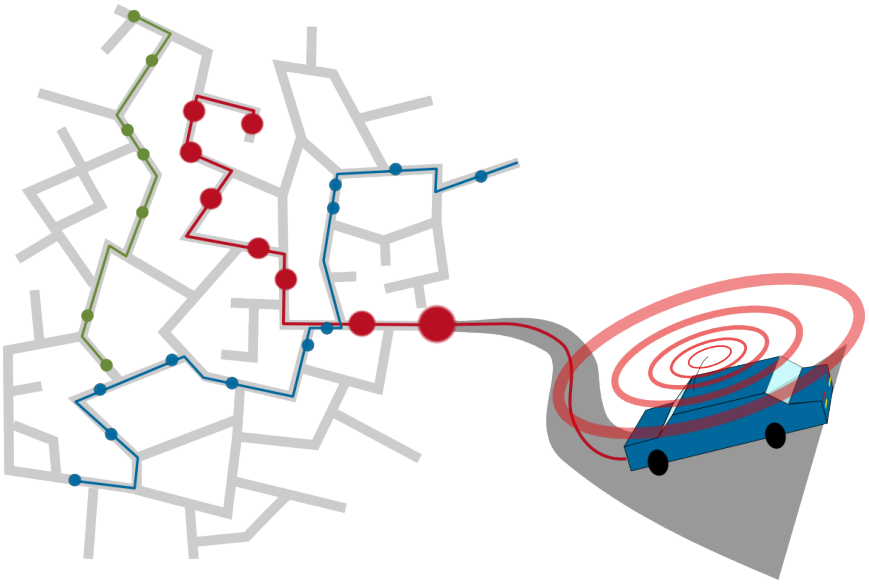


Figure 11.8: Vehicular Traces

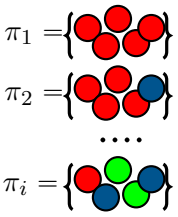


Figure 11.9: Unlinkability Hypotheses: Set of Partitions of a Set

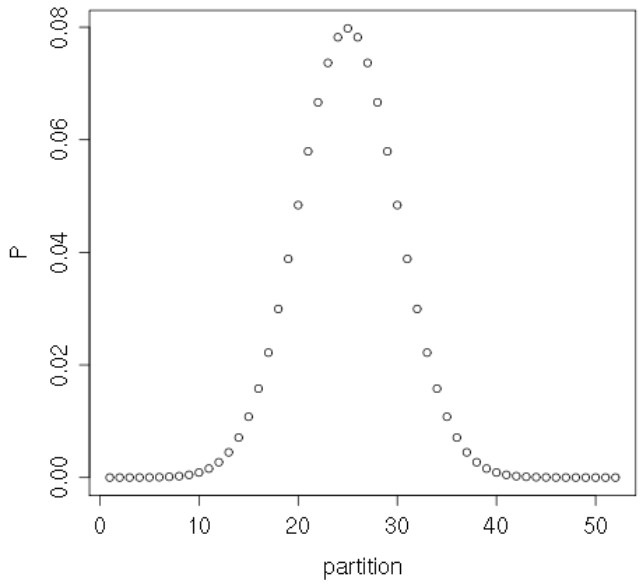
Consider a scenario of mobile devices (here it is vehicles) that - during their travels - emit messages containing positional information. Let's assume that these messages, aside from the positional information, is completely devoid of identification. Anybody overhearing some of these messages gains a set of position samples denoted M here.

The objective of an attacker is to partition this set according to their - unknown - senders. This means that the attacker reconstructs the traces of the mobile devices.

Positional Messages

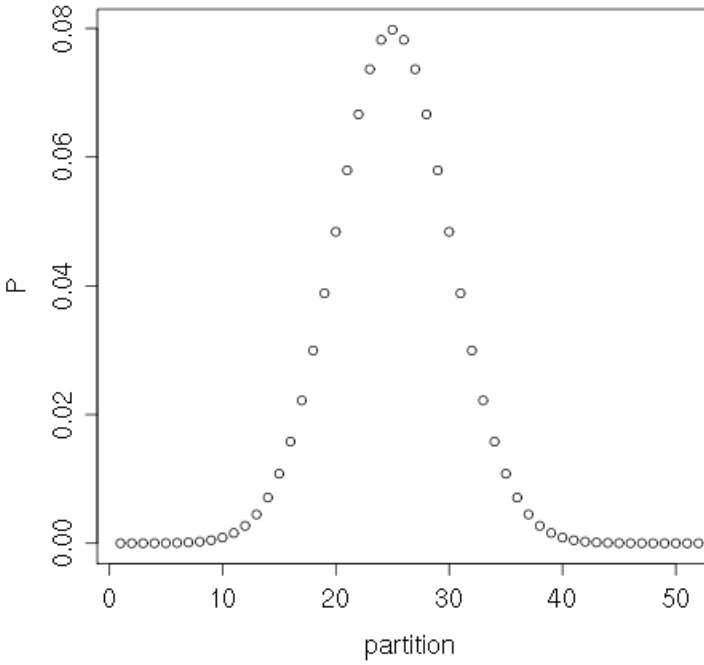
generates: Sample Points $M =< m_1, m_2, \ldots, m_n >$

Set Partitions: $\Pi =< \pi_1, \pi_2, \ldots, \pi_k >$, **Attacker's View** Probability Distribu-



tion $P : \Pi \rightarrow [0, 1]$

Question: *Amount* of Unlinkability/Disability of Attacker



Probability Mass Distribution(Example)
Degree of Unlinkability (Steinbrecher '01), (Franz '07)[3mm] $D(P) = \frac{H(\Pi)}{H_{max}(\Pi)} = 0.084$[4mm] Problem: Different Distributions can have same D.o.U.[3mm]

Measures Certainty of Attacker Doesn't Measure **Consistency**

Partition Difference

Unlinkability measures have been researched much later than anonymity measures. The first distinguished work on this problem by Sandra Steinbrecher und Stefan Köpsel adapted the Degree of Anonymity onto the set of all partitions of the set of items of interest. A partition of the set can be interpreted as the definition of an equivalence relation. Thus this Degree of Unlinkability provided a quantification of the certainty with which an attacker would choose a certain equivalence relation. [**steinbrecher03:˙model˙unlin**]

Consider a scenario of mobile devices (here it is vehicles) that - during their travels - emit messages containing positional information. Let's assume that these messages, aside from the positional information, is completely devoid of identification. Anybody overhearing some of these messages gains a set of position samples denoted M here.

The objective of an attacker is to partition this set according to their - unknown - senders. This means that the attacker reconstructs the traces of the mobile devices.

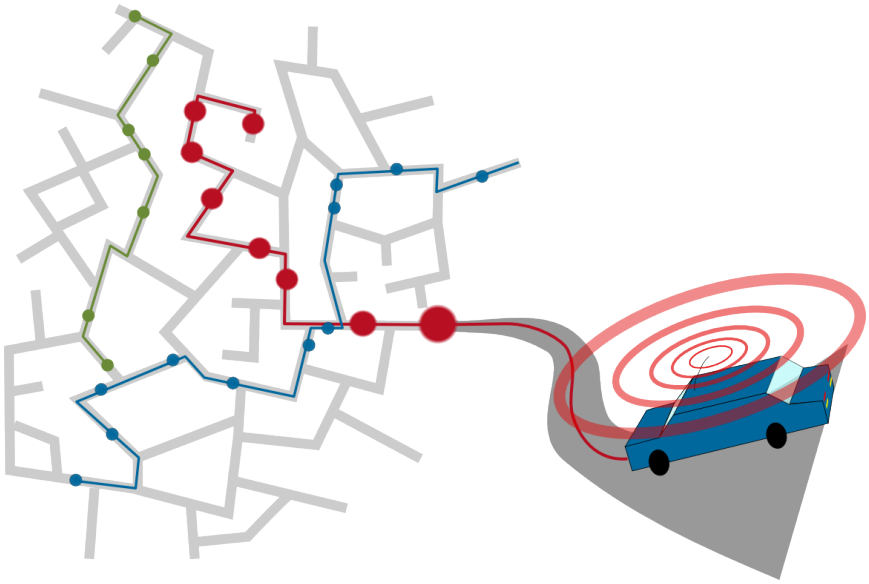


Figure 11.10: Vehicular Traces

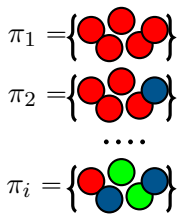
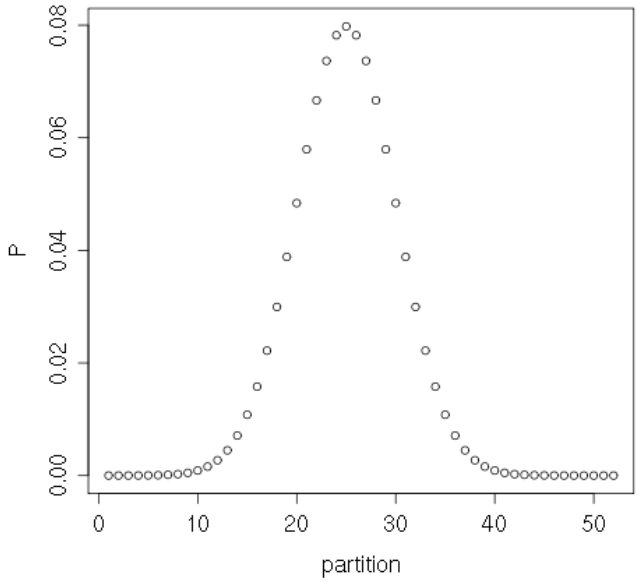


Figure 11.11: Unlinkability Hypotheses: Set of Partitions of a Set

Positional Messages

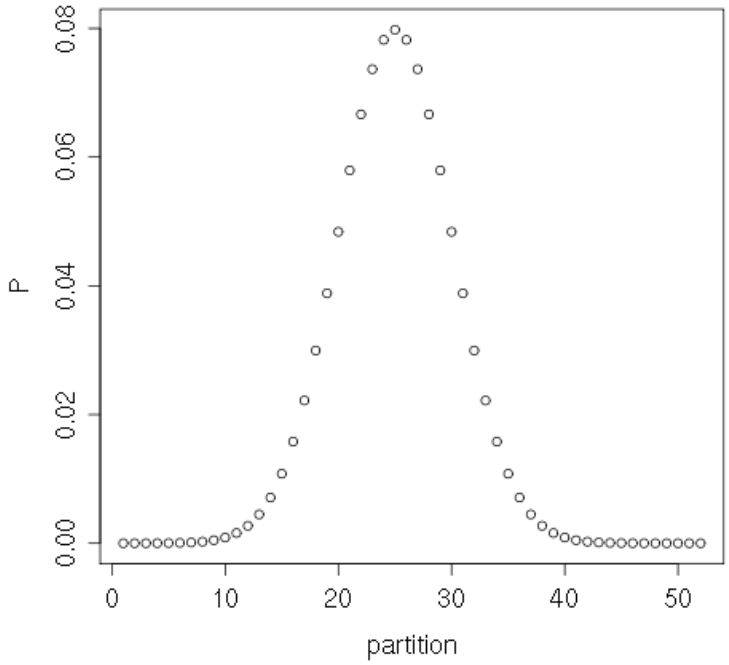
generates: Sample Points $M =< m_1, m_2, \ldots, m_n >$

Set Partitions: $\Pi =< \pi_1, \pi_2, \ldots, \pi_k >$, **Attacker's View** Probability Distribu-



tion $P : \Pi \rightarrow [0, 1]$

Question: *Amount* of Unlinkability/Disability of Attacker



Probability Mass Distribution(Example)
Degree of Unlinkability (Steinbrecher '01), (Franz '07)[3mm] $D(P) = \frac{H(\Pi)}{H_{max}(\Pi)} = 0.084$[4mm] Problem: Different Distributions can have same D.o.U.[3mm]

Measures Certainty of Attacker Doesn't Measure **Consistency**

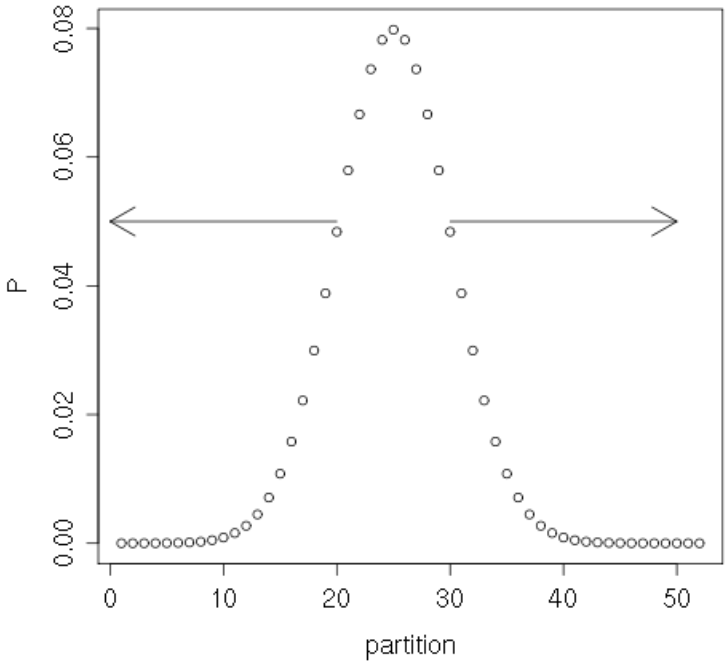
Expected Distance Unlinkability Measure

As a follow up I proposed to measure the expected error of an attacker as measure of his success.

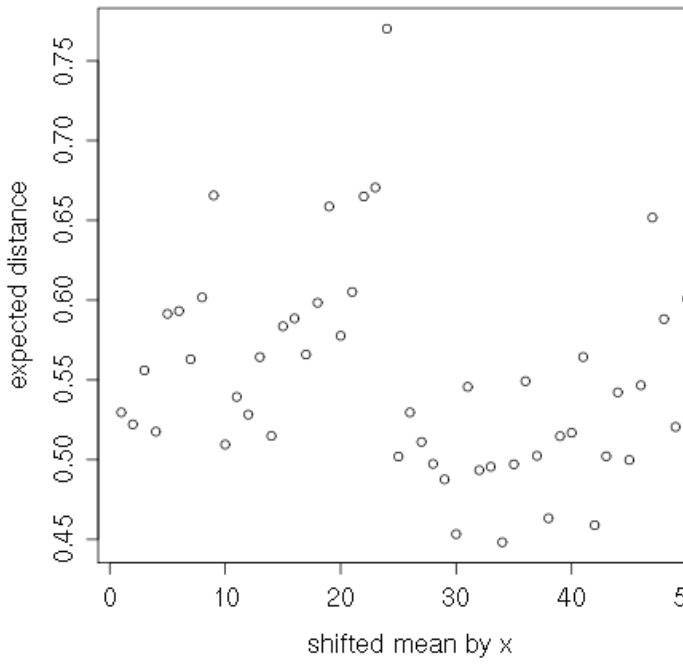
Expected Distance Unlinkability Measure[10mm]

$$ed(\pi) = \sum_{\pi \in \Pi} P(\pi) \cdot \delta(\pi, \tau)$$

τ: Reference Partition



Probability Distribution(Example)



Expected Distance $ed = \sum_{\pi \in \Pi_M} P(\pi) \cdot \delta(\pi, \tau)$

11.5 Privacy in the Internet

11.6 Mix Networks

Mix networks are overlay networks that route communication through multiple mix nodes in order to disguise the relation between sender and recipient. A mix node scrambles the relation between incoming and outgoing pakets by shuffling order, delaying tranfer and recoding. Mix nodes further have a function comparable to routers.

Adversaries are primarily distinguished as being external observers (*outsider attacks*) or being part of the mix network (*insider attacks*). An insider controls one or more mix nodes of the mix network. An outsider observes the communication of the network and is further classified by the degree of observed communication relative to all communication[3].

Mix Proxy Networks

- Sender Anonymity
- Redirection of Communication
- Breaking correlation of incoming and outgoing
- Hidden Services

Mix Nodes:

- Store-and-forward proxy
- Input: encrypted packets
- Delay packets
- Reorder packets
- Recode (e.g. unify lenght)

[3]For quite a while now the *global passive observer*, i. e., an adversary that observes all network communication, has been assumed to be unrealistic. Political events in 2013, related to the whistleblower Edward Snowden have taught the world that the unrealistic part has been the attribute *passive*. Global observation seems quite within the grasp of a few large organisations.

Attribute	Value	Rating
Cookies	Your browser does not store any cookies.	good
Authentication	Your unique ID: 322690120	bad
Cache (E-Tags)	Your unique ID: 832639264	bad
HTTP session	unlimited	bad
Referer	Original: Websites may see from which other website you come from!	medium
Signature	8ab3a24c55ad99f4e3a6e5c03cad9446 (Firefox)	medium
User-Agent	Mozilla/5.0 (X11; Linux i686; rv:10.0.5) Gecko/20100101 Firefox/10.0.5 Iceweasel/10.0.5	bad
Language	en-us,en;q=0.5	medium
Content types	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	medium
Encoding	gzip, deflate	good
Do-Not-Track	protected	good

[http://ip-check.info]

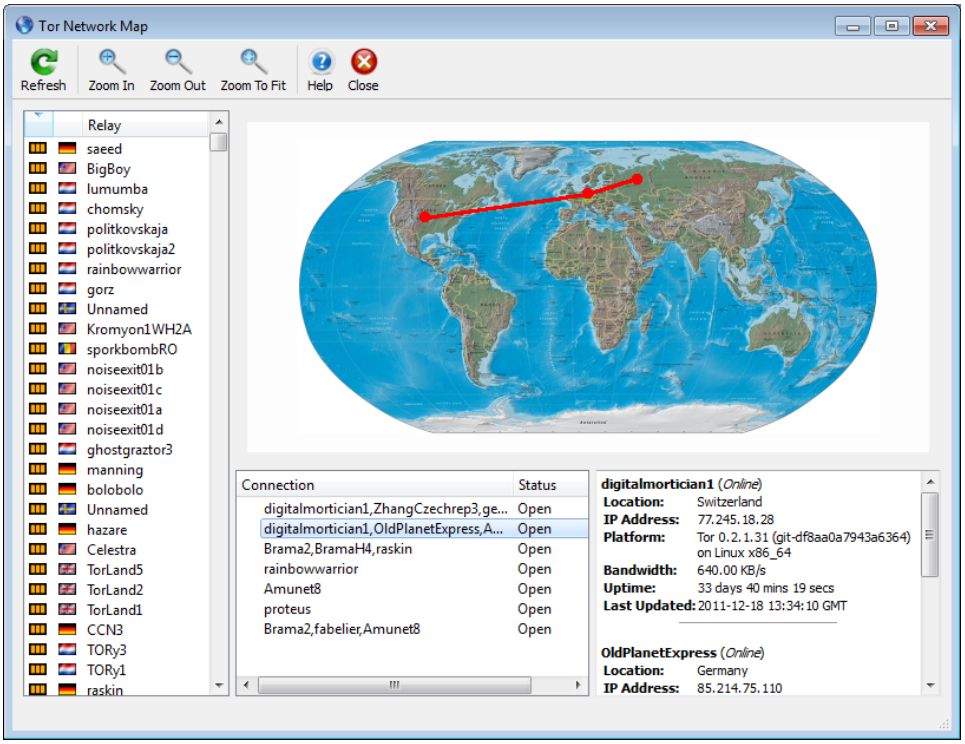
JavaScript	JavaScript is activated! (Version: 1.8)	medium
Plugins	Found 2 plugins.	medium
Mime types	Found 2 mime types that your browser supports.	medium
Tab name	"window.name" is traceable. Your unique ID: 7965368	bad
Tab history	There are 6 pages in your tab history.	medium
Screen	1680 x 1050 pixels, 24 bit color depth	medium
Browser window	838 x 1029 pixels, 838 x 921 pixels (inner size), Zoom: 100%	medium
Browser bars	MenuBar PersonalBar StatusBar ToolBar ScrollBars LocationBar	good
Local storage	Local storage is disabled or not supported by your browser.	good
Browser type	Mozilla/5.0 (X11) 20100101/20120605133945 Netscape (en-GB)	medium
System	Linux i686 Linux i686 (Thu Jul 05 2012 02:25:38 GMT+0200 (CEST))	medium
Fonts	36 installed fonts have been found on your computer.	bad
Browser history	Protected.	good

[http://ip-check.info]

11.6.1 How Tor Works

The Onion Router, better known by its abbreviation *Tor*, is a proxy network that provides a relatively high level of anonymisation on the network layer. Tor encapsulates each message within a TCP connection in multiple layers of encryption, that are successibly shed/applied, while a packet is passed from proxy to proxy. No individual proxy and no external observer gains knowledge about the identity of both endpoints of the connection from the protocol data.

- variable data relay paths
- world-wide distributed relays
- independent relay controller



Tor[4] can be assumed to be the most widely used mix network. Tor aims to hide the relation between sender and recipient for TCP-based communication. It is not providing unobservability against network-level observers.

The client establishes a *circuit* by selecting three proxy servers from the Tor directory servers and looking up their public keys. It then encapsulates the packets in layers of encryption so that each individual proxy is the recipient of one encrypted packages which contains the (encrypted) packet that is to be send to the next proxy.

[4] `https://www.torproject.org/`

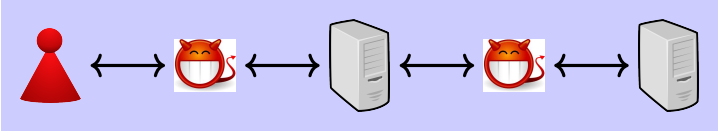
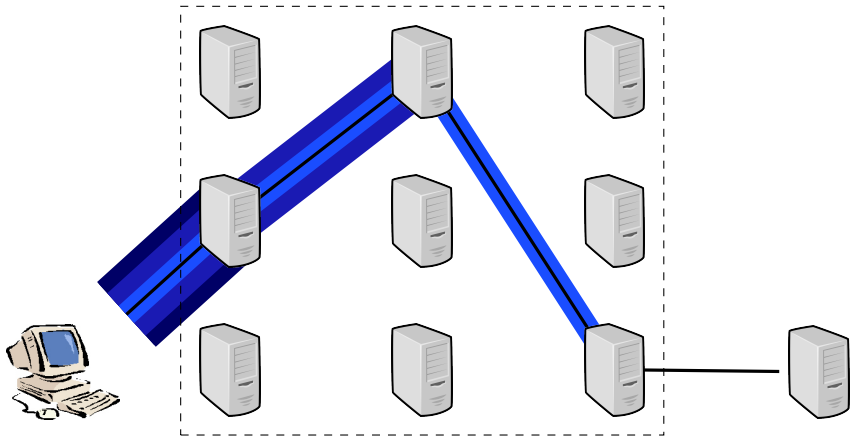


Figure 11.12: Adversary evesdrops on first and last relay



Thus, to express it more formally, the following messages are send on the Internet:

$$
\begin{aligned}
C \rightarrow P_{i_1} &: to : P_{i_1}, \{to : P_{i_2}, \{to : P_{i_3}, \{to : S, Data\}_{K_{P_{i_3}}}\}_{K_{P_{i_2}}}\}_{K_{P_{i_1}}} \\
P_{i_1} \rightarrow P_{i_2} &: to : P_{i_2}, \{to : P_{i_3}, \{to : S, Data\}_{K_{P_{i_3}}}\}_{K_{P_{i_2}}} \\
P_{i_2} \rightarrow P_{i_3} &: to : P_{i_3}, \{to : S, Data\}_{K_{P_{i_3}}} \\
P_{i_3} \rightarrow S &: to : S, Data
\end{aligned}
$$

Where[5] C and S are client and server within the TCP connection represented as $Data$. P_{i_j} represents the j-th proxy/relay within circuit i and $K_{P_{i_j}}$ the according public key used to encrypt the message enclosed within braces.

Proxy-Knowledge

A single proxy within the Tor network only has information about the previous and the next hop in the circle (which is controlled by the client).

1. adversary eavesdrops first and last relay
2. user sends his identity/IP in plain
3. adversary controls client

[5] Notation is adapted from [**abadi96prudent**].

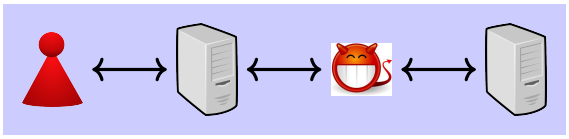


Figure 11.13: Adversary evesdrops last, plain IP in payload

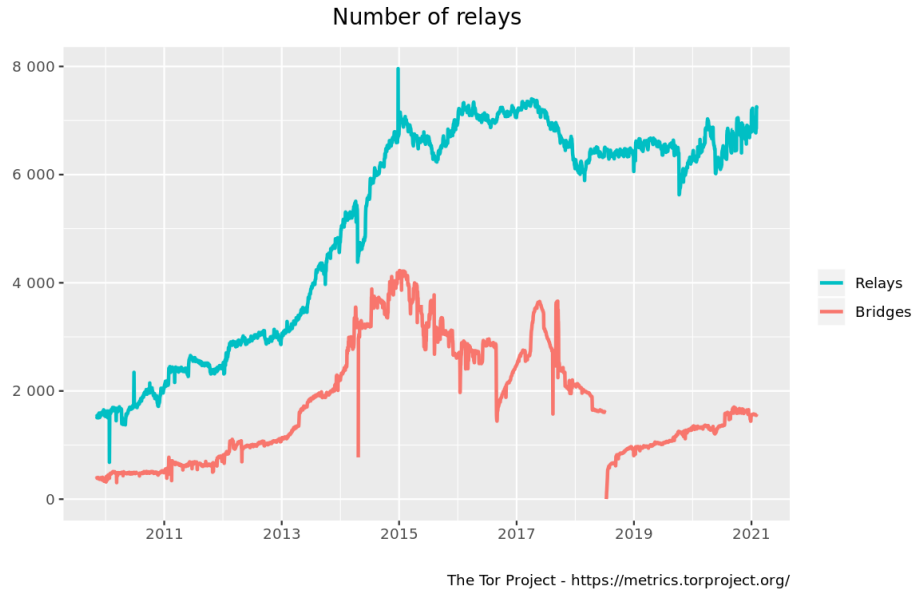


Figure 11.14: Number of Tor Relays [data source: `http://metrics.torproject.org`]

More Information About Tor

- ≈ 7000 relays
- 2 Million Users
- 500 GBit bandwidth

11.7 Database

A database is a structured collection of data with operations for storage and retrievals. The concept of databases also can be used as a generalisation for any kind of storage and retrieval of data and in this context to model concepts of database privacy.

Privacy in databases means to control the amount of information that can be learned from a database while allowing to retrieve defined information.

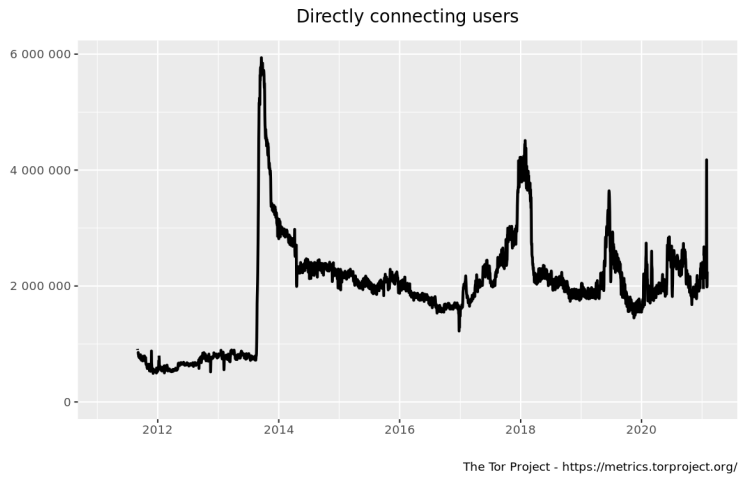


Figure 11.15: Number of Tor Users [data source: `http://metrics.torproject.org`]

- Controlled Information Disclosure
  - Anonymised Database
  - Database Queries

11.7.1 Relational Databases

Relational databases allow access to data using a relational algebra to retrieve information. Relational algebra allow to require data based on its relation to some expression, e. g., all birthdays of people living in (i. e., related to) a given area. Results of a relational query are sets of data or aggregates thereof.

Concerns with anonymity in databases is most often related to medical databases, which have been anonymised for a long time before it became a topic of research, by removing the names of individuals. In 1997 Latanya Sweeney published her PhD Thesis in which she could de-anonymise a set of anonymised medical records — distributed by hospitals for medical research — by comparing it to — essentially — a telephone book. [**Sweeney97ComputationalDisclosureControl**]

We distinguish between *sensitive attributes* that should not be disclosed for an individual, e. g., the diagnosis in a medical file. And *quasi-identifiers* that commonly are used to select records. Quasi-identifiers form an equivalence class with respect to a query, i. e., all records that match the query are in one equivalence class with respect to this query and all records that don't match are not equal with respect to that query.

The distinction into sensitive and non-sensitive attributes is disputed insofar as the conflict potential of an attribute highly depends on the context where it is

1	47677	29	Heart Disease
2	47602	22	Heart Disease
3	47678	27	Heart Disease
4	47905	43	Flu
5	47909	52	Heart Disease
6	47906	47	Cancer
7	47605	30	Heart Disease
8	47673	36	Cancer
9	47607	32	Cancer

Table 11.1: Example Relational Database (Source: [**Li07:t-Closeness:PrivacyBeyond**])

	ZIP	Age	Diagnosis
1	476**	2*	Heart Disease
2	476**	2*	Heart Disease
3	476**	2*	Heart Disease
4	4790*	$\geqslant 40$	Flu
5	4790*	$\geqslant 40$	Heart Disease
6	4790*	$\geqslant 40$	Cancer
7	476**	3*	Heart Disease
8	476**	3*	Cancer
9	476**	3*	Cancer

Table 11.2: Example Sanitised Relational Database with k-anonymity = 3 and l-diversity = 1

used.

Please consider the following privacy metrics as a basic introduction into the topic. A brief and concise introduction is given in [**Kelly:2011jr**]

k**-anonymity** means that the database must store or disclose data in a way that identifying attributes of each entry are within an anonymity set of size at least k.

l**-diversity** means that each equivalence class must at least contain l different values of the sensitive attribute. This shall prevent that an equivalence class contains only one value of the sensitive attribute.

t**-closeness** demands that the distribution of sensitive attributes for all equivalence classes must be similar, with respect to a distance metric for distributions.

ϵ**-indistinguisability** See Section 11.7.3.

11.7.2 Statistical Databases

Statistic databases provide data of statistical data, that means, instead of a set of rows with given attributes the output of queries are counts, averages and other statistical moments. We can distinguish two types of statistical databases: *pure statistical* which store only statistical data and *ordinary with statistical access* where users are only granted query rights restricted to statistical summaries.

A defining feature of statistical databases is a query function that maps onto statistical values, i. e., real values. Examples for query functions are “the sum of all members of a particular group” or “average income of a selection of individuals”.

11.7.3 Differential Privacy

Differential Privacy aims to quantify the amount of information that is leaked from a statistical database. Protection mechanisms based on differential privacy strive to keep the amount of leaked information below a leakage indicator ϵ by adding random noise to the answer to queries. Differential privacy is inspired by understanding of strength in cryptography, where results sometimes are required to be *statistically close* to be considered indistinguishable. [**Dwork06differentialprivacy**, **Dwork06calibratingnoise**]

Take the following example from the paper above. Given you provide a statistic of womens' body height by nationality? What does this disclose about Terry's body height? Not much. But if an attacker knows a further detail, for example the amount by which Terry's height differs from the average, then the attacker would gain precise information from the request.

- Statistic of womens height by nationality
  - Provides average body height
  - Additional Info: Terry is 2 in shorter than avrg. women
  - This provides a precise number of Terry's body height

Solution: Measure that does not depend on attacker context.

Thus a protection objective should be to protect privacy independent from the auxiliary knowledge an attacker might have. Or, at least have a measure for the amount of disclosed information independent from attacker context.

ϵ-Indistinguishability

A transcript t is a possible result of a database query consisting of a query Q and an answer a. For multiple subsequent queries a transcript is a sequence of

Figure 11.16: Absolute value of logarithm

queries and answers $[Q_1, a_1, Q_2, a_2, \ldots, Q_n, a_n]$. $\mathcal{T}(x)$ is the random variable from the set of all possible transcripts resulting from one or more queries on a database x. Then ϵ-indistinguishalbility is defined in [**Dwork06calibratingnoise**] as follows.

Definition 11.7.1 (ϵ-indistinguishability)**.** "A mechanism is *ϵ-indistinguishable* if for all pairs $x, x' \in D^n$ which differ only in one entry, for all adversaries $\mathcal{A}$ and for all transcripts t

$$\left| \ln \left(\frac{Pr[\mathcal{T}(x) = t]}{Pr[\mathcal{T}(x') = t]} \right) \right| \leqslant \epsilon.$$

In the introduction Cynthia Dwork roughly describes the idea behind the measure in Defininiton 11.7.1:

> "a privacy mechanism is ϵ-indistinguishable if for all transcripts t and for all databases x and x' differing in only one row, the probability of obtaining transcript t when the database is x is within an $(1 + \epsilon)$ multiplicative factor of the probability of obtaining transcript t when the database is x." [**dwork06calibratingnoise**].

Absolute Logarithm Value

To get an intuition on the behaviour of ϵ-indistinguishability it is useful to know the involved functions. Figure 11.16 shows the graph of the absolute value of the logarithm function used in epsilon undistinguishability. $abs.log$ briefly touches 0 at 1 on the horizontal axis, which means that the probabilities of two databases x and x' are equal (i. e., the quotient is one). The graph increases in both directions to infinity as the probabilities to produce the same transcript become increasingly different.

Example: Noisy Sum

Query functions that adhere to Def 11.7.1, i. e., stay below the threshold ϵ, are considered to be privacy preserving. An example, given in [**dwork06calibratingnoise**],

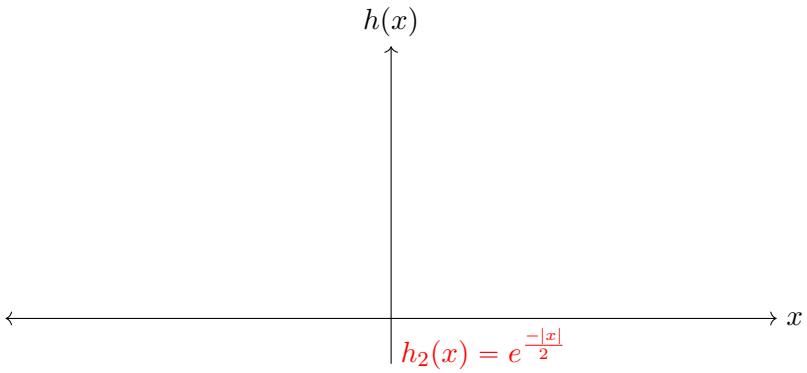


Figure 11.17: Laplace Distribution, with mean zero.

is "the sum of 1's in x" with added noise. The query function with Laplace-distributed noise can be defined as:

$$\mathcal{T}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i + Y, \quad \text{where } Y = Lap(1/\epsilon). \tag{11.2}$$

Where $Lap(b)$ has probability density function[6]

$h(x) \propto e^{\frac{-|x|}{b}}$.

The additional noise guarantees that the quotient

$\frac{Pr[\mathcal{T}(x)=t]}{Pr[\mathcal{T}(x')=t]} \leqslant e^{\epsilon|f(x)-f(x')|} \leqslant e^{\epsilon}$ is well below e^{ϵ}. Remember x and x' differ at exactly one entry, thus $|f(x) - f(x')| \leqslant 1$.

Why is it important to see this.

Laplace Distribution

Dwork shows that it is sufficient to include noise to the query function that follows the Laplace Distribution. This is very handy in terms of usability as the distribution has a slim peak and a steep dropoff, meaning that the noisy results often are very close to the actual result. The Laplace distribution, as shown in Figure 11.17, is symmetric (mirrored around the y-axis) and roughly hyperbolic. It often describes situations where the random value is modelling the difference between random variables, e. g., variance of a value over time.

The density function of the Laplacian distribution centred on zero is $\frac{1}{2\sigma}e^{-x/\sigma}$ where σ describes the spread of the distribution. A large value will produce more wide-spread noise.

[6] You will find that the Wikipedia-article differs by a factor, which is unimportant here.

Sensitivity of Query Functions

Assume that queries in a transcript are functions of the form $f : D^n \rightarrow \mathbb{R}^m$. There is a simple way to determine the amount of noise that is sufficient for ϵ-indistinguishability by way of the sensitivity of a function.

Definition 11.7.2. Sensitivity Sensitivity of a query function f is the smallest value $S(f)$ that is larger than the $\mathcal{L}_1$ distance for any two databases x, x' differing in only one row:

$$||f(x) - f(x')||_1 \leqslant S(f)$$

Privacy Preserving Queries

We use the Laplacian distribution to generate noise. Assume that you define your query function in a way that it adds a random variable Y to the result. Then we kno that for any y, y' drawn from that distribution $h(y)/h(y') \leqslant e^{|y-y'|/\sigma}$. But that means that all distorted query results (of the same query) are close to each other, or furthermore, that the transcripts are similar.

Or, as is proposed in [**dwork06calibratingnoise**]:

> For any query function $f : D^n \rightarrow \mathbb{R}^d$, the following mechanism is ϵ-indistinguishable:
> $SAN_f(x) = f(x) + (Y_1, \ldots Y_d)$, where Y_i are drawn from identically and individually distributed from the Laplacian distribution $Lap(S(f)/\epsilon)$.

add laplacian noise

11.8 OSN and Privacy

In a certain sense the whole Internet can be seen as a social network. But more specific the term herein is understood as a service that provides tools and webspace to publish a profile with explicitly established relations to other users, e. g., “friends”. An OSN thus is a graph of related user-profiles. Based on the relations an OSN may utilize informations of users differently for individual users, especially with respect to visibility of informations based on graph-distance of users.

- Attacker Model
  - OSN Provider/Database Access
  - Network Observer
  - Web Scraper
  - ...

- Attacker Objectives
  - Link Identities/Uncover Multi-Accounts
  - Reveal Domain-crossing Links
  - Copy/Crawl Database
- But: User want to share data
- Data-Protection vs. Usability

11.8.1 Profile Obfuscation

If you cannot run or hide, can you, at least, lead the attackers astray by providing false information in your profile?

- Nondescriptive Identifiers
  - Common Names (not Random Strings)
    - e.g. James Johnson, not cl34all2 unless. . .
    - But name is rather uninteresting
  - Misleading Fiendship Connections
    - e.g. accept all incoming friends
    - e.g. randomly send friend requests
    - But original graph is subgraph.
  - Fake Attributes
  - e.g. wrong age, hobbies,. . .
  - But our friends

Profile as masquerade:

- Fool observer/data-miner/clustering
- Creation of “harmless” profile
- Streamlined Profile:
  - Hiding within the masses.
  - Looks like everyone, writes like everyone

11.9 Privacy Mirrors

Privacy Mirrors are a Privacy as Practise technique that aims at enabling users to intervene and re-negotiate data usage boundaries. [**nguyen2002privacymirror**]

Exercises

Ex. 16 — Analyse further hazards of identification that are not covered by simply using Tor.

Contents

11.10 Threat Modelling

Wir behandeln Angriffsmodellierung hier ausschließlich mit dem Ziel den Schutz gegen Angriffe zu verbessern. Modelle helfen uns hierbei insbesondere bei der Identifizierung von Schwachstellen. Die Anwendung von Angriffsmodellen ist, in der Regel, die Durchführung von Bedrohungsanalysen (*Risk Identification*). Eine Bedrohungsanalyse soll Schwachstellen eines gegebenen Systems aufdecken.

Im Rahmen eines Sicherheitsprozesses, wie dem in Abb. 12.4 gegebenen ISO/IEC 31000 skizzierten Sicherheitsmanagementprozesses, ist eine Bedrohungsanalyse der erste Schritt des *Risk Assessment*.

In der Kombination können Bedrohungen und Auswirkungen zu einer Abschätzung des Risikos werden. Die Risikobewertung erlaubt zielgerichtete Auswahl effektiver Gegenmaßnahmen.

- Identifikation von Schwachstellen
- Quantifizierung des Risikos
- $Risk = Eintritt \times Schaden$

Gewünschte Eigenschaften:

- Quantifizierbarkeit: Eine quantitative Bewertung des Risikos erfordert, dass Bedrohungen möglichst formal und einheitlich formuliert werden.
- Vollständigkeit: Modellierung soll helfen einen möglichst großen Anteil der existierenden Schwachstellen aufzudecken. Vollständige Sicherheit ist nicht erreichbar, strukturierte Angriffsmodellierung soll helfen keine wesentlichen Teile in der Analyse zu übersehen.
- Kommunikation: Durch eine einheitliche Sprache lassen sich Schwachstellen einfacher Kommunizieren.

- Dokumentation: Die Formalisierung der *Risk Identification* dokumentiert die Ergebnisse und ermöglicht so weitere Verfeinerung der Analyse in späteren Prozessen.

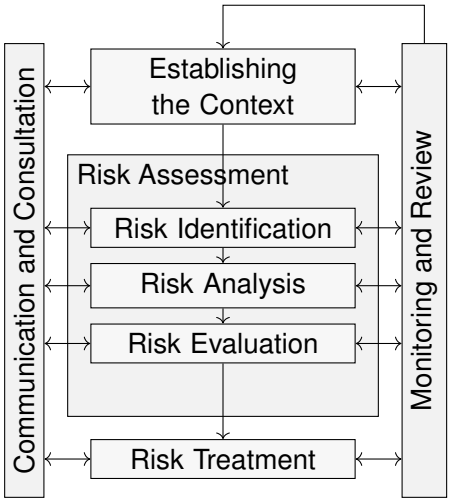


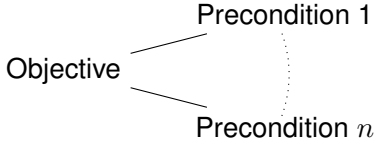
Abbildung 11.18: Risikomanagementprozess nach ISO/IEC 31000

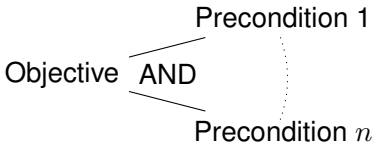
11.10.1 Attack Trees

Attack Trees are a (if not the) fundamental method to systematically explore attack vectors and preconditions to attack objectives. They can be used to explore different possibilities to reach an objective, quantify the relative costs of different attack path or the minimum costs of an attacker to breach a target. The term is probably coined by Bruce Schneier in 1999, but the concept is well known from Fault-Tree-Analysis. [8]

Angriffsbäume (*Attack Trees*) stellen die Abhängigkeiten von Angriffsschritten, oder Zwischenzielen, zur Erreichung jeweils eines Angriffszieles dar. Sie unterstützen die Identifizierung von Angriffswegen und bereiten damit die Quantifizierung des Angriffsrisikos vor. Ziel der Erstellung von Angriffsbäumen sollte es sein alle wesentlichen Angriffspfade darzustellen.

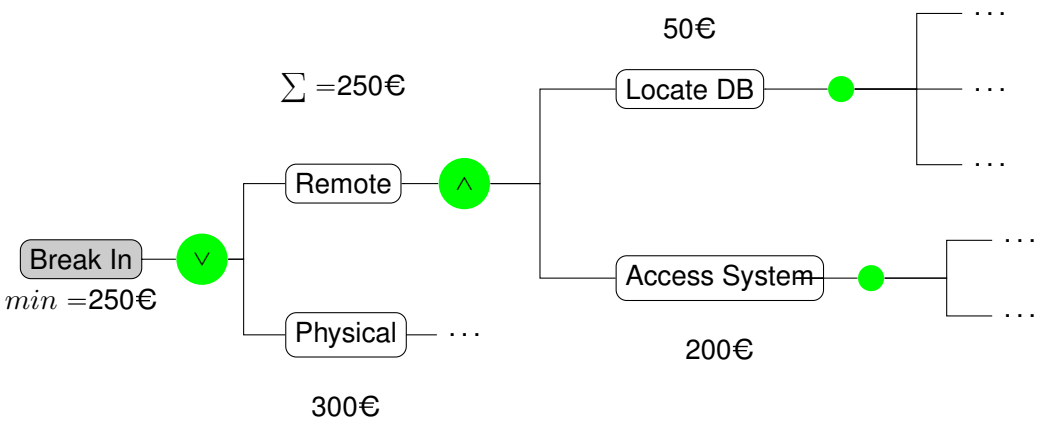
The lowest leafs of attack trees can be attributed with estimators of complexity or probability of realisation of sub-goals. It is very common to use monetary quantifiers to represent complexity, i. e., costs, of an attack. Costs can easily be accumulated upwards to derive the overall costs of an attack.





- Tree of preconditions
- Every precondition is sub-objective
- Preconditions: AND or OR
- Node annotation, e. g., complexity/cost

Costs are one factor to estimate the probability of an attack. Other influencing factors are motivation of an attacker, e. g., by estimating the profit an attacker can make from a successful attack. Aggregating this into a single measure provides a frequency or general probability of an attack.



To estimate the risk, it is necessary to also calculate the expected damage that has to be compensated if a successful attack strikes.

A very common criticism is, that already the estimation of the damage is tarnished with uncertainty. Especially costs that are produced from bad publicity are difficult to estimate.

11.10.2 Elemente der Angriffsmodellierung

Um Angriffe modellieren zu können, muss die Modellierungssprache die folgenden Elemente vorsehen.

- System
  - Abhängigkeiten

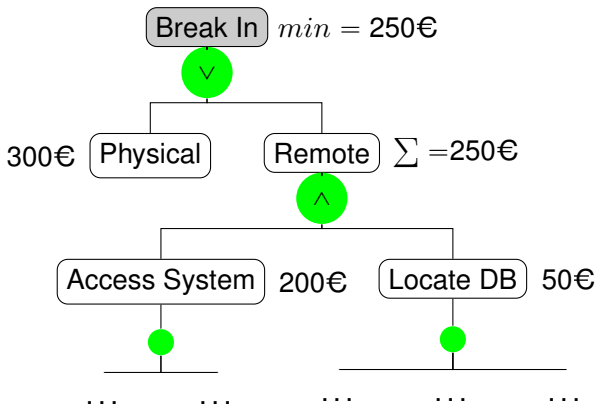


Figure 11.19: Example Attack Tree

  - Schutzziele
- Angreifer
  - Fähigkeiten
  - Motivation/Ziele
- Interaktion
  - Angreiferaktionen
  - Ergebnisse
- Bewertung

Ich möchte hier unterschiedliche Stufen der Angriffsmodellierung unterscheiden.

- Angriffspfade:
  - Attack Tree [**Saltzer**, **Schneier**] siehe Abb. **??**
  - Attack-Defence Tree [**Schweitzer**]
- Algorithmisch:
  - Attack Execution Graph (AEG) [**LeMay2011aeg**]
- Kausale Systemmodelle:
  - Cyber Security Modelling Language (CySeMoL)erlaubt die Beschreibung des untersuchten Systems als probabilistisches relationales Modell. Ein Modell beschreibt die kausalen Zusammenhänge eines Systems. Aus den kausalen Zusammenhängen werden in der Auswertung Angriffspfade generiert. Cyber Security Modelling Language

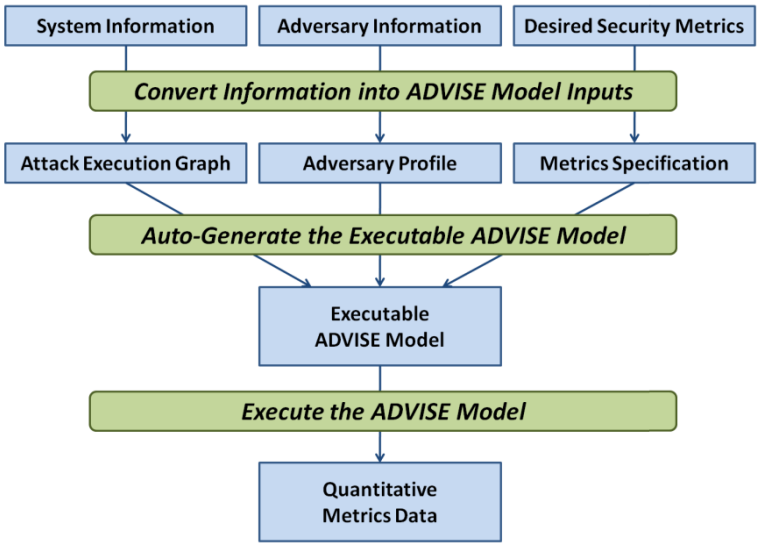


Figure 11.20: ADVISE methodology [**LeMay2011aeg**]

(CySeMoL) automatisiert damit die händische Entwicklung des Angriffsgraphen.

- Digital Twin: Zukunftsmusik ist die Modellierung des kausalen und zeitlichen Verhaltens, welche die Auswertung realer Echtzeit-Interaktion erlaubt. Die Modellierung soll eine effizientere Auswertung erlauben, um, zum Beispiel Angriffe anhand ihrer Auswirkungen auf das System anhand der Auswertung des digitalen Zwillings zu erlauben.
- Guideword-Aufzählung
  - STRIDE [**stride**]
  - SecHAZOP

11.10.3 Attack Execution Graphs

In diesem Abschnitt führe ich die Angriffsmodellierung mittels Attack Execution Graphs (AEGs) modellieren den Ablauf von Angriffen als attributierte Flußgraphen.

In den Beispielen dieser Vorlesung werden wir AEG vornehmlich für die Identifizierung von Angriffspfaden und die Bestimmung von Maßnahmen benutzen. Darüber hinaus sind AEG aber insbesondere für die Risikoabschätzung gedacht.

Um geeignete, zielgerichtete Gegenmaßnahmen zu identifizieren muss zunächst das betrachtete System, sowie die zu untersuchenden Angriffsziele eingegrenzt werden. Dabei müssen insbesondere auch die Anforderungen in Bezug auf die

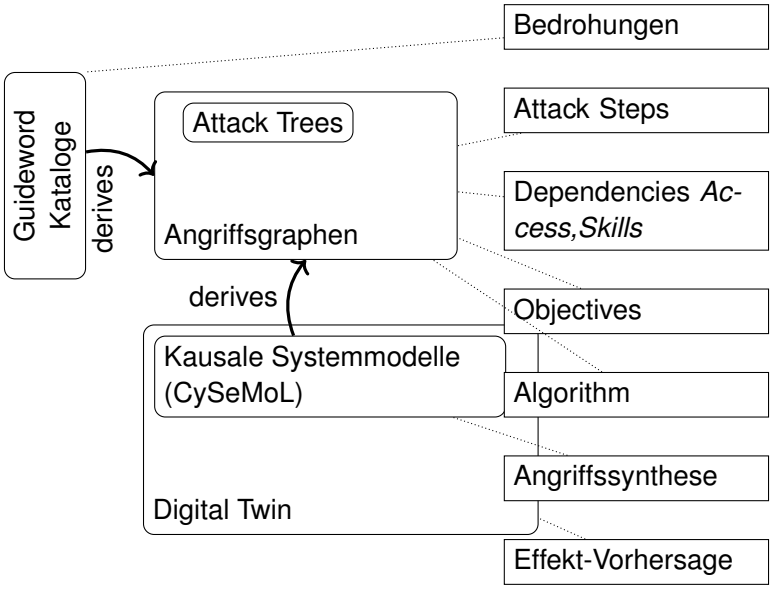


Figure 11.21: Zusammenhänge unterschiedlicher Konzepte der Angriffsmodellierung

Detailtiefe der Angriffsschritte mit den Auftraggebern abgestimmt werden. Darauf basierend wird, in einem kreativen Prozess, der Angriffsgraph zur Beschreibung aller Angriffswege modelliert. Die beiden Schritte sollten so lange zyklisch wiederholt werden, bis sich ein konsistentes Bild ergibt. Die Detailtiefe der Angriffsschritte muss dabei konstant mit den Anforderungen an das Ergebnis abgeglichen werden.

Dieser Prozess, dargestellt in Abbildung 11.22, wird manuell durchgeführt erfordert umfassende Expertise bezüglich des betrachteten Systems, von Angriffstechniken und realistischen Angriffsmöglichkeiten. Er lässt sich ungefähr auf die Phasen *Establishing the Context*, *Risk Identification* und *Risk Treatment* des Risikomanagementprozesses nach ISO/IEC 31000 abbilden, überspringt dabei die Phasen *Risk Analysis* und *Risk Evaluation*.

AEG quantifizieren Risiko basierend auf Angreiferklassen (*Threat Agent Classes*). Angreiferklassen definieren Stereotype Angreifer, die sich in Bezug auf verfügbare Resourcen, Ziele und Einschränkungen unterscheiden. In Abbildung 11.23 ist ein Ergebnisbeispiel für die weiter unten genauer definierten Angreiferklassen gezeigt. Die Quantifizierung des Risikos erfolgt pro Angreiferklasse durch die Berechnung einer Time-to-Compromise (TtC).

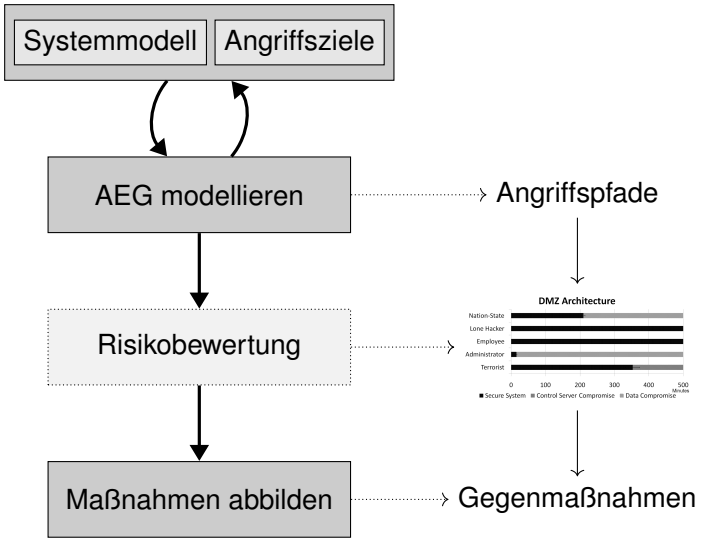


Figure 11.22: Prozess zur Erstellung von Angriffsgraphen

AEG Komponenten

Wir unterscheiden im folgenden zwischen *Angriffsschritten* und *Zustandsknoten*. Angriffsschritte sind *Attack Step*-Knoten und Zustandsknoten sind alle anderen, namentlich *Skill*, *Knowledge*, *Access*, und *Goal*.

Angriffsschritte beschreiben, ähnlich den Transitionen in Petri-Netzen, Zustandsübergänge. Wobei ein Zustandsübergang dann erfolgen kann, wenn alle Eingangsbedingungen erfüllt sind. Das Ergebnis eines Zustandsübergangs ist die Erfüllung aller Ausgangsknoten. Zum Beispiel die Erlangung eines Zugangs zu einem bestimmten Systemteil, dargestellt durch einen *Access*-Knoten.

AEG ein Graph mit Knoten $\langle A, R, K, S, G \rangle$, wobei die Elemente die folgenden Komponenten bezeichnen:

Attack Step $a_i \in A$ eine “Transition” während eines laufenden Angriffs.

Access $r_i \in R$ “**R**essourcen” die einem Angreifer verfügbar sind.

Knowledge $k_i \in K$ Wissen über die Architektur, Credentials, Prozesse, Codes,...

Skill $s_i \in S$ Fähigkeiten eines Angreifers, die nicht während des Angriffs gewonnen werden können, z.B. spezielle Programmierfähigkeiten.

Goal $g_i \in G$ Ziele, die für unterschiedliche Angreifer von jeweils eigene Wichtigkeit haben können.

.

Angriffsgraph: $\langle A, R, K, S, G \rangle$

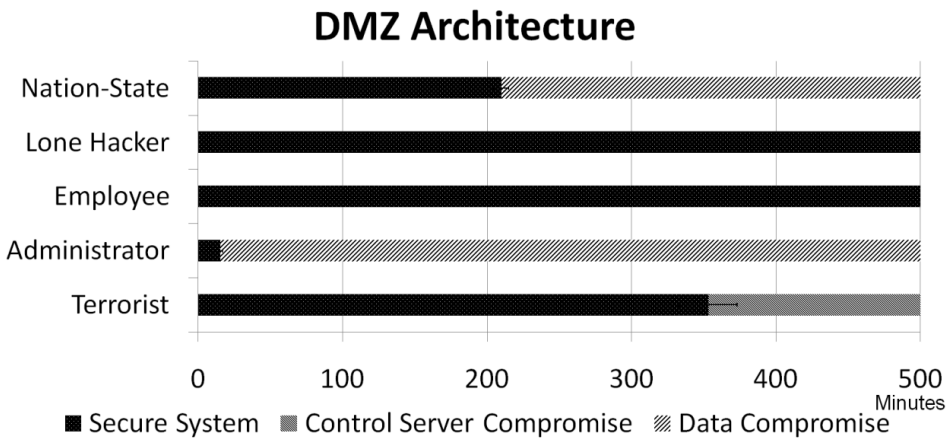


Figure 11.23: AEG-result, time-to-compromise for different threat agent models. [**LeMay2011aeg**]

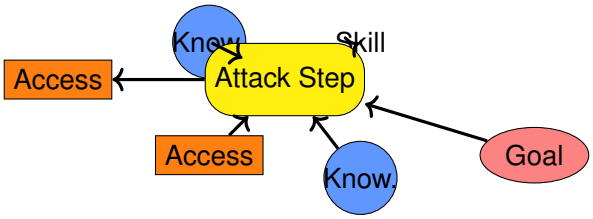


Figure 11.24: Knotenklassen und Kanten in AEG

Zu einem Graph gehören in der Regel auch die Kanten, welche die Knoten verbinden. In AEG geschieht diese Verknüpfung in den *Attack Step*-Knoten. Dabei wird ein *Attack Step* jeweils mit Vorbedingungen und Resultaten verknüpft. Vorergebnisse sind Knoten aus R, K, S. Resultate sind Knoten aus R, K, G. Die Interpretation ist, dass ein Angriffsschritt nur dann ausgeführt werden kann, wenn alle Vorbedingungen wahr sind. Als Ergebnis der Durchführung eines Angriffsschrittes gelten alle Resultate nach der Durchführung als erfüllt.

Ein Beispiel: Ein Ergebnis des erfolgreichen Abhören eines Passworts, zum Beispiel durch *Shoulder-Surfing* ist das Wissen (*Knowledge*) des Passworts. Kenntnis des Passworts ist eine Vorbedingung des Angriffsschrittes *Einloggen*, durch welchen der Angreifer Zugriff (*Access*) auf den Computer mit den Rechten und der Identität des abgehörten Nutzers erlangen kann. (Siehe Abb. 11.31)

Grapheigenschaften von AEG

AEG sind bipartite Graphen, wodurch die herausgehobene Bedeutung der *Attack Step*-Knoten betont wird, welche die eine der beiden Knotenmengen exklusiv bilden. Die Richtung der Kanten definiert die Fließrichtung der Ausführungsrei-

henfolge von Angriffsschritten. In der Regel sind AEG nicht zyklisch. (Abb. 11.25)

In der informellen Modellierung ist es aber gelegentlich sinnvoll ähnliche Angriffstechniken gegen unterschiedliche Systemkomponenten zusammenzufassen. In der formalen Analyse sind zyklische Strukturen nicht sinnvoll, weil jedes Ergebnis nur einmal erreicht werden muss. (Anmerkung: gegebenenfalls machen zyklische Strukturen Sinn in der probabilistischen Modellierung, weil die wiederholte Ausführung eines Angriffsschritts die Erreichung eines Ergebnisses wahrscheinlicher macht. Formal kann z.B. laterale Ausbreitung allerdings nicht über zyklische Strukturen modelliert werden.)

Diskussion: Zyklische AEG

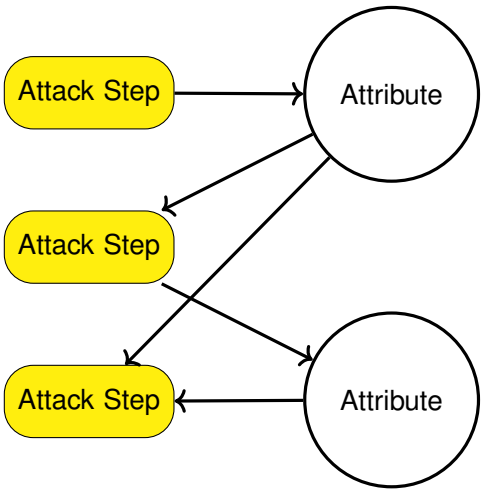


Figure 11.25: Die Knotenmengen in AEG sind paarig, d.h. es gibt Kanten nur zwischen Angriffsschritten und anderen Knoten. Es gibt keine Kanten zwischen *Skill, Knowledge, Access, Goal*-Knoten, und keine Kanten zwischen Angriffsschritten.

Attack Step

Angriffsschrittes stellen Zustandsübergänge in AEG dar, die beschreiben, wie ein bestimmter Graphenzustand $s \in X$ in andere Zustände übergehen kann. Die Rolle ist vergleichbar mit den Transitionen in Petri-Netzen. In AEG sind deshalb die Verbindungen und wesentlichen Attribute, insbesondere die Kanten eines AEG in den *Attack Step* definiert.

In einer vereinfachten Variante, definieren wir zunächst nur die Kanten des Graphen und inklusive der Zustände der Eingangsbedingungen eines *Attack Step*.

Ein vereinfachter Angriffsschritt ist ein Tuple:

$a_i =< B_i, O_i >$

wobei B_i die Vorbedingungen und O_i die Ergebnisse eines Angriffsschrittes definiert.

Preconditions $B_i : X \rightarrow \{True, False\}$

Outcomes O_i (finite set)

Bemerke: Ein Angriffsschritt beschreibt die Vorbedingungen als Abbildung aus einem gegebenen Auswertungszustand $s \in X$, auf den wir weiter unten eingehen werden. Die Ergebnisse *Outcomes* eines Angriffsschrittes werden über eine Teilmenge der Zustandsknoten beschrieben.

Um zu verstehen, wie das Modell funktioniert müssen wir das ganze mal praktisch anwenden. Angenommen wir haben den Auftrag eine Schwachstellenanalyse für ein Wasserwerk zu machen. Bestellt ist eine Red-Team Penetrationstest und zur Vorbereitung sollten wir uns schon einmal die gängigsten Angriffswege aufzeichnen.

Modellierung von Prozessabläufen

Die Definition der Angriffsschritte liefert die Syntax der (noch nicht attributierten) AEG. Als nächstes müssen wir verstehen, wie wir gängige Flusssemantiken abbilden.

Der grundlegende Flussgraph, der über den einzelnen Schritt hinausgeht, ist eine Sequenz von zwei Schritten. Die Verkettung erfolgt dadurch, dass aus einem Angriffsschritt die gleichen Ergebnisse resultieren, wie im nächsten Angriffsschritt als Vorbedingungen gefordert werden. Die leeren runden Knoten in Abbildung 11.26 stehen jeweils für eine oder mehrere Bedingungen (*Skill, Knowledge, Access*).

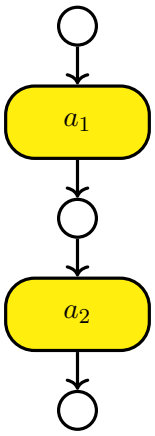


Figure 11.26: Building a Sequence of Attack Step

Die nächste betrachtete Flusssemantik ist die Verzweigung eines Flusses in

zwei unterschiedliche Pfade. In AEG bedeutet dies entweder, dass ein Angriffsschritt mehrere Ergebnisse produziert, die in disjunkte Pfade münden (Verzweigung A in Abb. 11.27). Oder, dass eine Vorbedingung zwei unterschiedliche Angriffsschritte ermöglicht, welche dann wieder in disjunkte Pfade münden (Verzweigung B in Abb. 11.28).

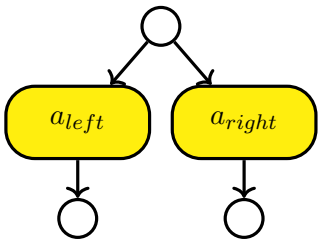


Figure 11.27: Verzweigung durch zwei mögliche Angriffsschritte

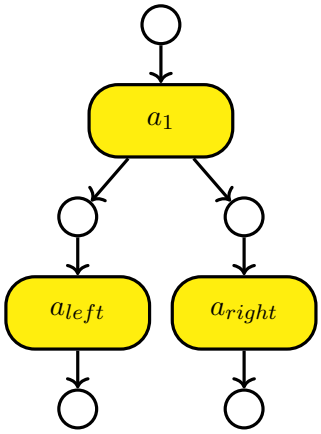


Figure 11.28: Verzweigung durch zwei Ergebnisse

Als letzten Punkt müssen wir schauen, wie Disjunktion und Konjunktion von Vorbedingungen in AEG dargestellt werden können. (Abb. 11.29 und 11.30)

Beispiel Wasserversorgung

$a_i = \langle B_i, T_i, C_i, O_i, Pr_i, D_i, E_i \rangle$

Preconditions $B_i : X \rightarrow \{True, False\}$

Outcomes O_i (finite set)

Probability of Success $Pr_i : X \times O_i \rightarrow [0, 1]$, $\sum_{o \in O_i} Pr_i(s, o) = 1$

Time Required $T_i : X \times \mathbb{R}^+ \rightarrow [0, 1]$

Cost $C_i : X \rightarrow \mathbb{R}^{\geqslant 0}$

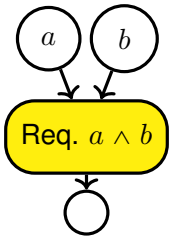


Figure 11.29: Konjunktion der Vorbedingungen eines Angriffsschrittes ist die normale Behandlung der Bedingungen

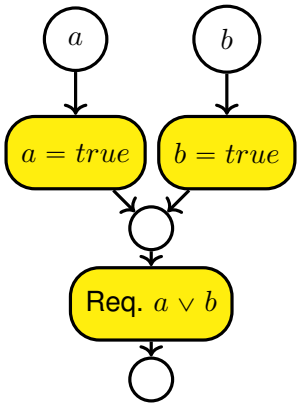


Figure 11.30: Disjunktion der Vorbedingungen eines Angriffsschrittes ist die normale Behandlung der Bedingungen

Detectability $D_i : X \times O_i \to [0, 1]$

State Transition $E_i : X \times O_i \to X$. Describes the state transition if outcome O_i occurs.

The state does not contain the state of outcomes, “only” of preconditions.

[**LeMay2011aeg**]

Graph Zustand

Bevor wir uns allerdings den Attack Steps zuwenden können, müssen wir uns kurz die Auswertung eines AEG anschauen. Dies geschieht indem der durch den AEG definierte Angriffsweg schrittweise nachvollzogen wird. Dadurch besteht eine Auswertung aus einer Sequenz von Zuständen der *Access*, *Knowledge*, und *Goal*-Knoten. Der Zustand eines Knotens ist ein boolscher Wert, der beschreibt ob eine Vorbedingung oder ein Ziel erfüllt ist.

Ein spezifischer Zustand ist also ein Tripel der zugehörigen Zustandsmengen.

Evaluation of a given AEG

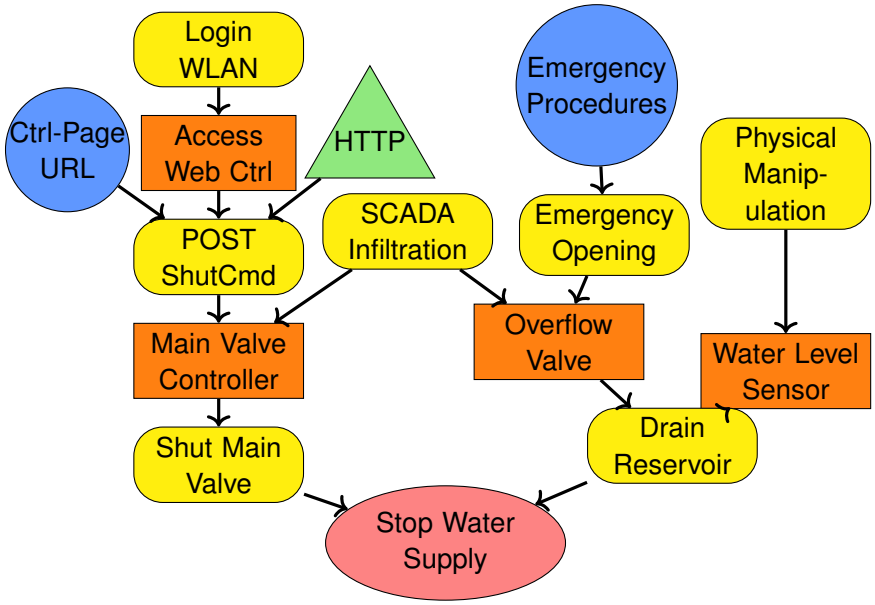


Figure 11.31: Modellierung der Angriffswege die zur Abschaltung der Wasserversorgung führen können als AEG

Model State: $s \in X$, $s = \langle R_s, K_s, G_s \rangle$ Note that “skill” is not part of the state, i. e., considered immutable.

Zustandsübergang

Ein Zustand wird beschrieben durch die Mengen der erfüllten Zugriffsdomänen $R_s \subseteq R$, Wissensblöcke $K_s \subseteq K$, und Ziele $G_s \subseteq G$.

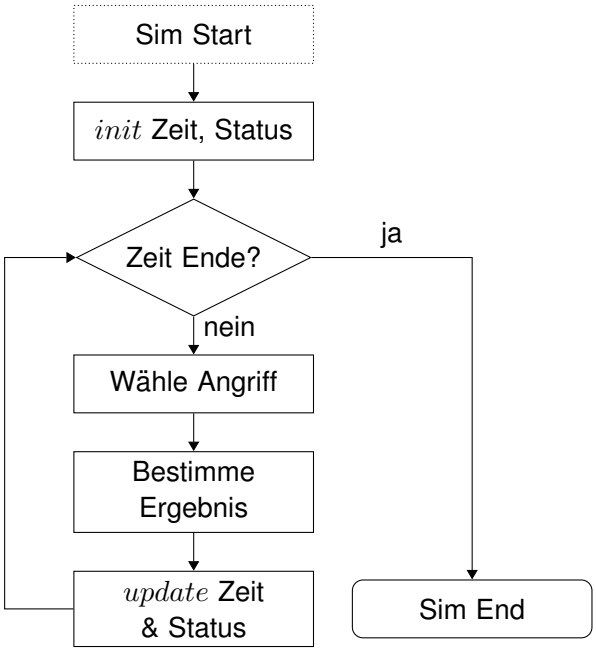
Zustand: $s = \langle R_s, K_s, G_s \rangle$

1. Auswertbare *Attack Step*: $A_s = \{a_i \in A | B_i(s) = True\}$
2. Bewerte Attraktivität der *Attack Step*
   - Zufall
   - Short-Sighted Attacker

Die Auswertung eines Graphmodells erfolgt iterativ, wobei jeweils ein einzelner Angriffsschritt pro Iteration ausgewählt wird. Durch die Anwendung des Angriffsschritts wird ein Zustandsübergang $s_i \rightarrow s_{i+1}$ definiert. Wobei in Zustand s_i alle Anforderungen des Zustandes erfüllt sein müssen.

Der neue Zustand s_{i+1} ergibt sich durch die Auswertung der (probabilistischen) Outcomes.

Figure 11.32: Informelle Beschreibug des Ablaufs der Auswertung



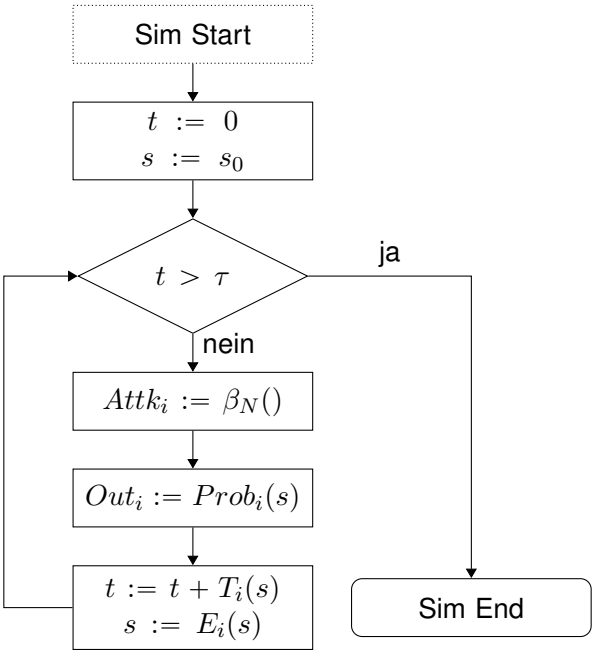


Figure 11.33: Schematische Darstellung des formalen Ablaufs

τ Simulation Endzeit

β_N Angriffsauswahlfunktion welche den Angriffsschritt mit der höchsten Attraktivität (aus Sicht des Angreifers) auswählt.

$Prob_i$ Wahrscheinlichkeit des Outcomes i im Zustand s

T_i

Threat Agent Model

In AEG threat agents are modelled as *Attacker Profile*, defining properties and objectives of an attacker, and by the evaluation function for selection of the next attack step in a simulation.

Attacker Profile

Individuelle Angreiferklassen werden in AEG durch wesentlich durch die Priorisierung der Ziele Kosten, Ertrag (*Payoff*) und Entdeckungsrisiko (*Detectability*) definiert. Dadurch werden unterschiedliche Einschränkungen modelliert. Zum Beispiel verfügt ein Angreifer, der durch einen Nationalstaat finanziert wird über vergleichsweise hohe finanzielle/technische Resourcen, was durch ein relativ niedriges Gewicht für die Kosten w_C repräsentiert wird. Beispiele sind in Tabelle **??** gegeben.

Weitere Unterscheidungsmerkmale sind die Fähigkeiten der Angreiferklassen, die für jeden *Skill* des verwendeten AEG angegeben werden. Ausserdem hat die Erreichung der unterschiedlichen Ziele G einen Wert der subjektiv für einzelne Angreifer ist.

Neben den Fähigkeiten und Zielgewichten die ein Angreifer mitbringt "starten" unterschiedliche Angreiferklassen an unterschiedlichen Punkten. Ein interner Angreifer zeichnet sich, zum Beispiel, dadurch aus, dass er bereits Zugriff auf bestimmte Komponenten eines Systems hat oder über mehr internes Wissen verfügt als ein externer Angreifer. Dies wird dadurch dargestellt, dass jede Angreiferklasse von einem anderen initialen Startpunkt s_0 in die Simulation einsteigt.

Darüber hinaus bestimmt der Wert N die weite des Planungshorizont eines Angreifers. Das bedeutet, dass ein Angreifer die erwarteten Resultate für Gewinn, Kosten und *Detectability* für N Schritte in die Zukunft in die Attraktivitätsbewertung des nächsten Angriffsschrittes einfließen lässt.

Ein Angreifer wird durch ein Tupel:

$< s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N >$ repräsentiert.

s_0 initialer Startpunkt. Unterschiedliche Angreifer starten ihre Angriffe von unterschiedlichen Punkten aus. Ein *interner Angreifer* hat ggF. schon mehr Zugriffsrechte.

Adversary	w_C	w_P	w_D
Nation-State	0,01	0,4	0,59
Lone Hacker	0,2	0,4	0,4
Terrorist	0,05	0,8	0,15
Employee	0,4	0,5	0,1
Administrator	0,4	0,5	0,1

Table 11.3: Beispiele für Angreiferprofile [**LeMay2011aeg**]

$L: S \rightarrow [0,1]$ (*Skill*) für jeden *Skill*-Knoten.

$V: G \rightarrow \mathbb{R}^+$ Subjektiver Gewinn in Geldäquivalent für die Erreichung der Angriffsziele.

$w_C, w_P, w_D \in [0,1]$ individuelle Gewichtung der Priorisierung (*Attractiveness*) der Kosten C (*Cost*), Gewinnerwartung P (*Payoff*), und Heimlichkeit D (*Detectability*) bei der Auswahl des nächsten Angriffsschritts.

U_C, U_P, U_D *Utility*-Funktionen zur Abbildung der Prioritätskriterien auf ein Einheitsinterval $[0,1]$.

N Planungshorizont eines weitsichtigen Agenten in Zeit oder Angriffsschritten, je nach genutztem Auswertungsalgorithmus.

Short-Sighted Adversary

Der *Short-Sighted Adversary* bewertet die Attraktivität der möglichen nächsten Angriffsschritte nur anhand der erwarteten Kosten, Gewinne oder *Detectability* des einzelnen Schrittes. Essentiell hat er einen Planungshorizont von $N = 1$.

$$attr(a_i, s) = w_C \cdot C_i(s) + w_P \cdot P_i(s) + w_D \cdot D_i(s)$$

Die Auswahl der Gewichte für die unterschiedlichen Angreiferklassen sollte auf einer fundierten Analyse der Bedrohungslage beruhen. Die Angreiferdefinition ist wesentlicher Teil jeder Sicherheitsarbeit. Letztlich handelt es sich dabei in der Regel aber um subjektive Einschätzungen. Um die Frage nach den Möglichkeiten eines Angreifers etwas handhabbarer zu machen ist es hilfreich Angreifer in mehreren Kategorien zu definieren. Ein hilfreiches Werkzeug ist die INTEL Threat Agent Library [4, 3].

Long-Range-Planning Adversary

Der *Long-Range-Planning Adversary* bewertet die erwarteten Resultate aller möglichen Angriffsschritte für N Schritte im Voraus. Anders gesagt, er analysiert

einen N-stufigen, probalisitischen Zustandsbaum, ausgehend vom aktuellen Zustand s.

State Look-Ahead Tree (SLAT):

- Knoten: mögliche Zustände
- Kanten: Angriffsschritte
- Konstruktionstiefe: N
- Vereinfacht:

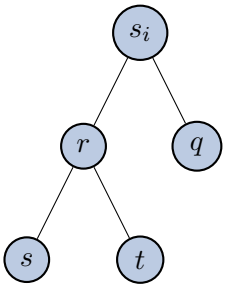


Figure 11.34: Zustandsbaum ausgehend vom aktuellen Zustand s_i

Graphgenerierung:

- Ermittlung der möglichen Angriffsschritte a_i in Zustand s
- Berechnung der möglichen Outcomes von a_i in Zustand s
- Berechnung der Ergebniszustände s_i
- Wdh. für alle Ergebniszustände bis zur maximalen Baumtiefe N

Komplexität: Exponentiell ($O(c^N R)$für einen positiven Wert c.

- Pro Suchtiefe multipliziert sich die Anzahl der betrachteten Knoten.
- Für kleine N berechenbar.

$$attr^N(a_i, s) = w_C \cdot C_i^N(s) + w_P \cdot P_i^N(s) + w_D \cdot D_i^N(s)$$

Kosten: Summe der Pfadkosten

$$C_i^N(s) = C_i(s) + \sum_{o \in O_i} \left(C_*^{N-1}(r) \cdot Pr_i(s, o) \right), N > 1$$

Gewinn: Summe der Gewinn der Blätter

$$C_i^N(s) = \sum_{o \in O_i} \left(P_*^{N-1}(r) \cdot Pr_i(s, o) \right), N > 1$$

Detectability: Produkt der summierten Pfadwahrscheinlichkeiten

$$D_i^N(s) = \sum_{o \in O_i} ((1 - (1 - D_i(s,o)) \cdot (1 - D_*^{N-1}(r))) \cdot Pr_i(s,o), N > 1$$

Wobei C_*^{N-1}, P_*^{N-1}, und D_*^{N-1} jeweils die Kosten, Gewinne, Detectability des Angriffsweges mit maximaler Attraktivität sind.

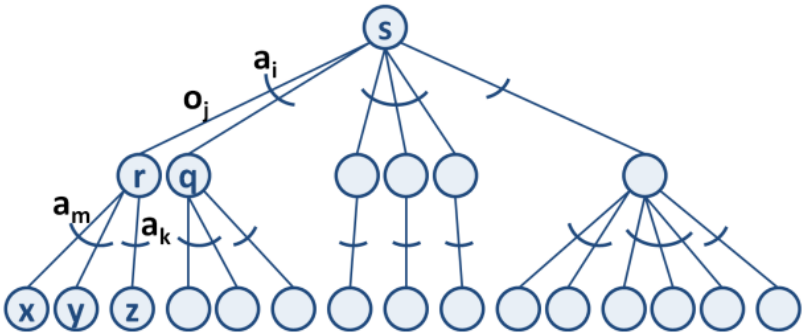


Figure 11.35: *State Look-Ahead Tree* ausgehend vom Zustand s

Weitere Beispiele

Petya/No-Petya 2017

Die Kette von Vorfällen in 2017, die übergreifend mit dem Wurm “Petya” verbunden werden bestehen einerseits aus einem RANSOMWARE-Angriff, und einem vermutlichen “Trittbrettfahrer”. Der Trittbrettfahrer, der auch als “No-Petya” bezeichnet wird, hat sich ähnlich verhalten wie der originale Schadcode. Eine Analyse des Codes hat allerdings ergeben, dass die Verschlüsselungsfunktion keine Entschlüsselung vorgesehen hat. (Abb. 11.36)

Double Pulsar

Der Exploit *Double Pulsar* wurde im *WannaCry* RANSOMWARE-Angriff genutzt. Er basiert auf einem Werkzeug das von der National Security Agency (NSA) entwickelt und von den *Shadow Brokers* “geleaked” wurde. (Abb. **??**)

11.11 Vulnerability Quantification

Exkurs:

Vulnerability Example OpenSMTPD[7]

[7] `https://packetstormsecurity.com/files/156137/OpenBSD-OpenSMTPD-Privilege-Escalation-Code-Execution.html`

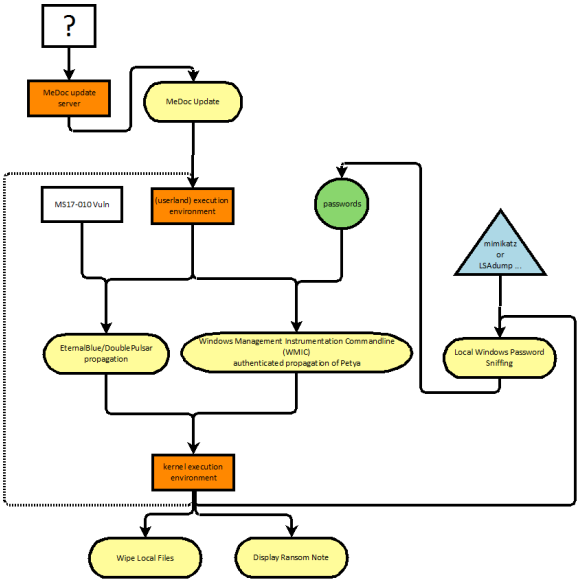


Figure 11.36: AEG describing the flow of the Petya Incidents 2017

This is an excellent example for failed input validation. The error is easily understooy once it is pointed out, the damage is most devastating and the solution can be found by novice programmers.

- CVE 2020-7247
- Attack Technique: CAPEC-88: OS Command Injection
- Weaknesses:
  - CWE 78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
  - CWE 88: Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')
  - CWE 20: Improper Input Validation
  - CWE 697: Incorrect Comparison
  - CWE 713: OWASP Top Ten 2007 Category A2 - Injection Flaws

From `smtp_mailaddr()`:

```
static int
smtp_mailaddr(struct mailaddr *maddr, char *line,
       int mailfrom, char **args, const char *domain) \{
```

```
        if (!valid_localpart(maddr->user) ||
            !valid_domainpart(maddr->domain)) \{
```

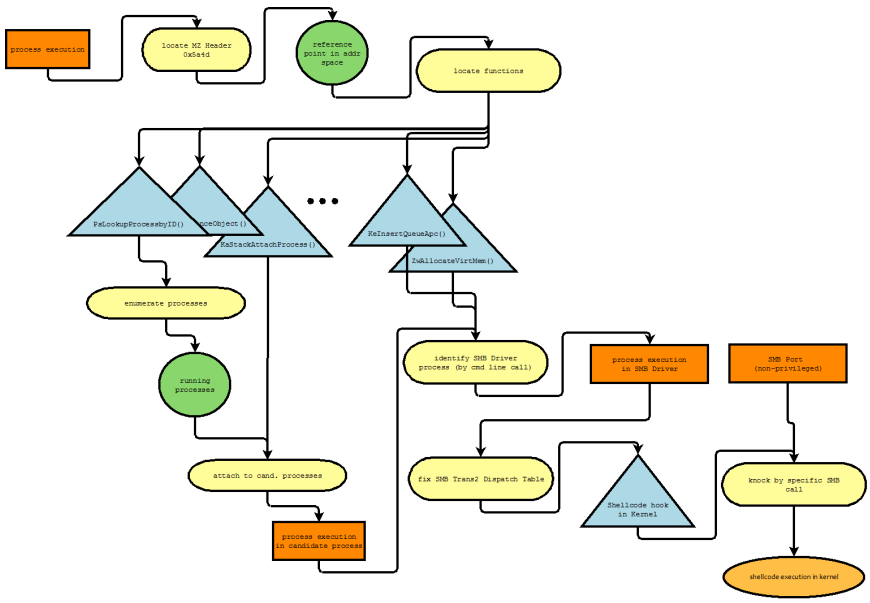


Figure 11.37: AEG for DoublePulsar Attack

```
                    if (maddr->domain[0] == '\0') \{
                            (void)strlcpy(maddr->domain,
                                domain,
                                 sizeof(maddr->domain));
                            return (1);
                    \}
                    return (0);
            \}
            return (1);
\}
```

The problem is, that the function returns 1 (= everything OK) in the case `user` of a `mail to`-address is malformed and `domain` is empty. The code in line 2230 “repairs” the domain part, but leaves the user part unmodified. A fitting weakness category from the Common Weakness Enumeration (CWE) would be *CWE-393: Return of Wrong Status Code*. But to repair the problem, the structure of the conditional branching has to be refactored to distinguish the cases of invalid user and domain and always fail (return 0) if any is invalid. Empty is a special case of invalid domain, but this must not lead to ignore an concurrently invalid user.

Local emails are delivered executing the value of `mda_command` on the shell.

MTA is called as:

```
execle("/bin/sh", "/bin/sh", "-c",
```

```
        mda_command,
        (char *)NULL, mda_environ);
```

default value is

`mda_command` ist constructed:

```
  asprintf(&dispatcher->u.local.command,
    "/usr/libexec/mail.local␣-f␣%%{mbox.from}␣%%{user.
       username}");
```

As the attacker is able to (almost) arbitrarily choose the value of `user.username`, is is the case, he can inject arbitrary command-code for the Shell.

Because the input validation for the username is effectively skipped, the exploit code can be almost arbitrarily be chosen. For example an attacker could execute a denial-of-service on the smtp service Netcat to SMTP-Server

```
HELO
REPT TO:someone@example.org
MAIL FROM:<;sleep 66;>
DATA
```

11.12 Attack Patterns

11.12.1 CAPEC

The Common Attack Patterns Enumeration and Classification (CAPEC) provides a hierarchically structured collection of attack techniques (i. e., attack patterns). The structure provides three different levels of abstraction. Different *Views* on Common Attack Patterns Enumeration and Classification (CAPEC), e. g., “by Mechanism” allow a flexible way of navigating the collection. CAPEC is the most comprehensive collection of attack patterns, which has incorporated most other collections like the WASC Threat Classification.

https://capec.mitre.org/

11.12.2 MITRE ATT&CK (ATT&CK) Framework

The MITRE ATT&CK (ATT&CK) Framework provides a tool for structured expression of TTP. MITRE ATT&CK (ATT&CK) is structured by the phases of the Cyber-Kill-Chain (CKC) in a matrix of attack patterns. It relates attack phases and steps to CWE and CAPEC with the intention to identify and distinguish the threat agent groups based on the TTP that they commonly deploy.

11.12.3 Threat Agent Classification

At the start of an attack there has to be an actor who has motivation and means to facilitate the attack. The effectivity of attacks thus depends to a large extend on the capabilities of that *threat agents* are able to activate. These capabilities can differ widely in type and extend, comprised of available knowledge, hardware, access and software.

The *INTEL Threat Agent Library* [4] provides a scheme to classify different types of attackers with respect to objectives, limits and ressources. In that way it provides a terminology to talk about different attacker models on a very high level.

Civil Activist

Radical Activist

Anarchist

Competitor

Corrupt Government Official

Cybervandal

Data Miner

Disgruntled Employee

Government Cyberwarrior

Government Spy

Internal Spy

Irrational Individual

Legal Adversary

Mobster

Sensationalist

Terrorist

Thief

Vendor

Access	Internal External
Outcome	Acquisition/Theft Business Advantage Damage Embarrassment Tech. Advantage
Limits (max)	Code of Conduct Legal Extra-legal, minor Extra-legal, major
Resources (max)	Individual Club Contest Team Organisation Government

Skills (max)	None Minimal Operational Adept
Objectives	Copy Deny Destroy Damage Take
Visibility (min)	Overt Covert Clandestine

11.13 Security Analysis Process

[5, pg. 226]

Threat and Risk Analysis are a fundamental part of the Security Development Lifecycle (SDL) Process (see Section **??**). There exists a wide variety of analysis concepts for threats and risks. A starting point for your own research could be the Master Thesis of Katrin Scholz. [**Schol2004EntwicklungeinerMethodik**]

A good analysis method should support the analysts in providing a complete list of realistic threats. It should allow to represent realistic and balanced estimations of risks.

There is no exhaustive standard on the actual execution of a penetration test. But there are a few more-or-less obvious things to consider. First, obviously, would be the preparation. You will need a few tools, you should establish a few processes and during an assignment there is an almost natural order of steps to follow. It might be of little surprise, that some things are very similar to the preparation of an actual attack. The main difference is, that you might be required to succeed as often as you can, are allowed to leave traces and, most important, should be protected from prosecution.

- Laboratory
  - Logging Facilities/Database
  - Attack Tools
    - (Virtual) Analysis Computer
    - Hardware depending on tasks
    - Code Analysis/Reverse Engineering Facilities
    - SW-Dev Toolchain
  - Demonstrators
- Training
  - All-You-Can-Eat
  - Build Demonstrators
  - Reverse Engineering Malware
  - Tools-of-the-Trade
- Report Templates

1. Assignment Scope
2. Preparation

3. Reconnaissance
4. Gain Access
5. Maintain Access/Extend Access
    - Most tests stop here
6. Cover Tracks (only in special assignments)
7. Report!

Get a Permission Slip/Contract!

- Objectives
- Target Systems
- Time Frame
- Mode of Operation
- Team Requirements
- Team Training
- Assemble Tools (see Kali Linux)
- Customise to Task
- Enumerate Interfaces
    - Scanning
    - Web-Scraping
- Identify System-components
    - Software Stack
    - Hardware
    - Legit Users
    - Application Context
- Research Known Vulnerabilities

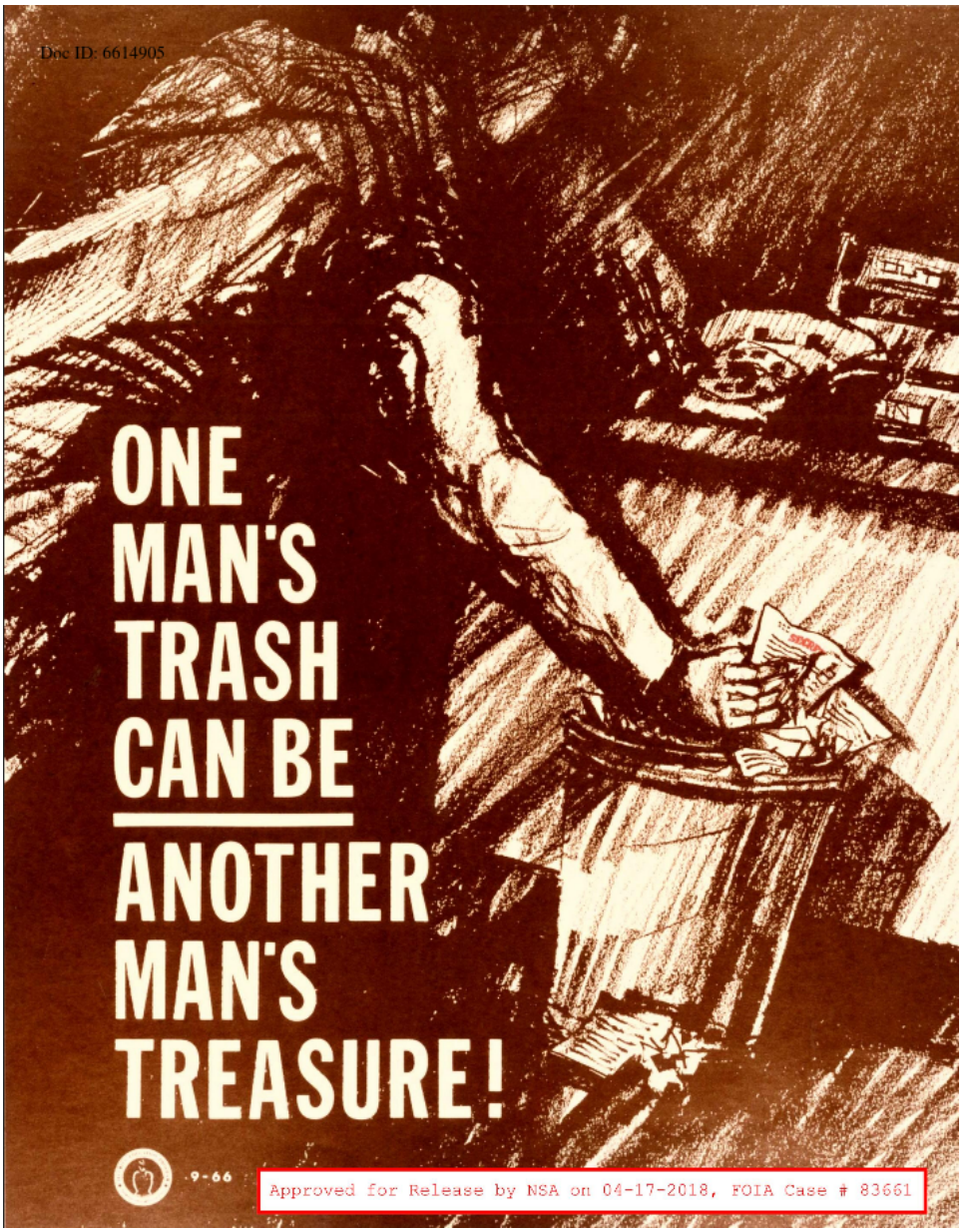
Reconnaissance Phase of a Pentest/Attack. This section is a note on the tools to introduce.

11.14 Introduction

Open-Source Intelligence (OSINT) describes all techniques to derive information on targets from publicly available sources. This includes technical infor-

mation systems, like Domain Name System (DNS), or scanning techniques as well as physical or social techniques.

- Intelligence from public sources
- (military term)
- “Digital Humphrey Bogart”



¡++¿

“Intelligence derived from publicly available information, as well as

other unclassified information that has limited public distribution or access." [The official Nato Terminology Database]

"(1) Open-source intelligence (OSINT) is intelligence that is produced from publicly available information and is collected, exploited, and disseminated in a timely manner to an appropriate audience for the purpose of addressing a specific intelligence requirement." [NATIONAL DEFENSE AUTHORIZATION ACT FOR FISCAL YEAR 2006, Subtitle D, a) 1)

11.14.1 Gap Analysis Method

1. What do I know?
2. What does it mean?
3. What do I need to know?
4. How do I find out?

Using Gap Analysis For Smarter OSINT (2020-03-15)[seen 2021-04-27]

As an example for this method I "stalked" the hull of a military vessel being tugged from Bremerhaven to Bremen.

On the 2021-04-26 the newly built hull of a military vessel was in the news for being transferred from Bremerhaven to Bremen to being tested in the water. (Alledgedly the harbour at the ship builder's was to shallow for the ship.) [Buten un Binnen, 2021-04-26, Warum dieses Kriegsschiff nach Bremen kommt] I read the news the morning after and was curious whether the vessel was still in transfer.

But, for a start, military vessels are among the first to not have their Automatic Identification System (AIS) switched on, and second that vessel was still a hull. Thus, it has not been successful to localize the vessel directly.

Let's follow the gap analysis method:

- What did I have? A picture of the vessel being towed by two tugs with a swimming crane attached — and a date. But I did not have the vessel in any shiptracking page. I did not have any ship names, mostly probably, because I did not look to deeply into the images.[8] I did, though, have the destination port of the convoy.
- What does it mean? The vesel could probably not be located directly, but I might be lucky with the other vessels in the convoy. The transport probably was still going on, thus, finding a suspicious combination of tugs and swim-crane, could be a hint.

[8] Well, this is an example and what would be the fun of it?

- What did I need to know? The location of a suspicious looking collection of vessels on the river between port of origin and destination.
- How do I find out? Looking at a marine-traffic-map, starting with the destination port, going down the river and marking all suspicious collections of tugs and cranes.

It turned out very quickly, that at the destination port a crane, looking very similar to the crane on the picture, was moored directly beside a tug named “Bremerhaven”. A second tug from Bremerhaven was moored right beside a free spot in that harbour. The military vessel was not to be seen, but given the suspicious free space beside the second tug, I would put my bet on the free spot. I concluded to not follow this through the end, and did not take my bicycle to verify my findings, after all, this is just an example and I have no reason to stalk this vessel.

11.15 Open-Source Intelligence (OSINT)-Techniques

OSINT Resources

11.15.1 Domain and Network Adresses

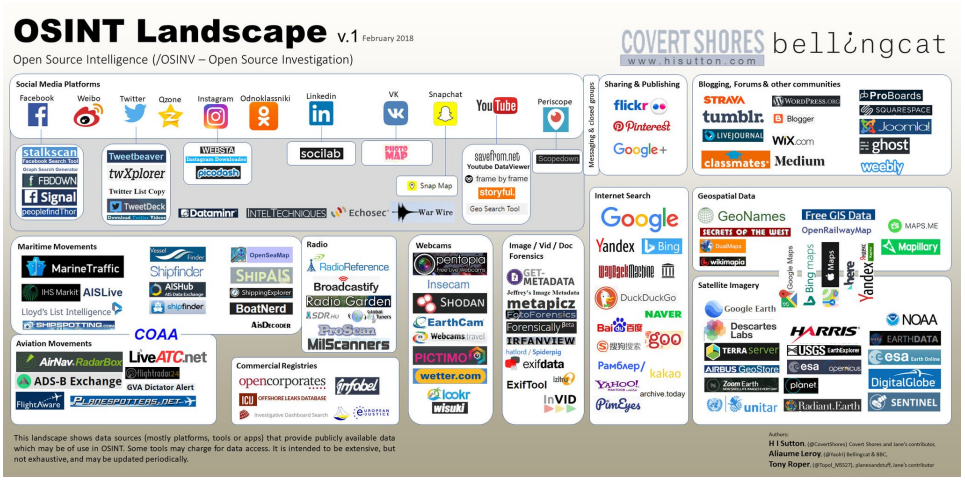
OSINT Domain and Network Adresses

11.15.2 Linked Data

So, you are collecting an awful lot of information on a specific target — and now you are struggling with gaining insights from it?

osint handling information

11.16 Tools



Tools to support an analysis:

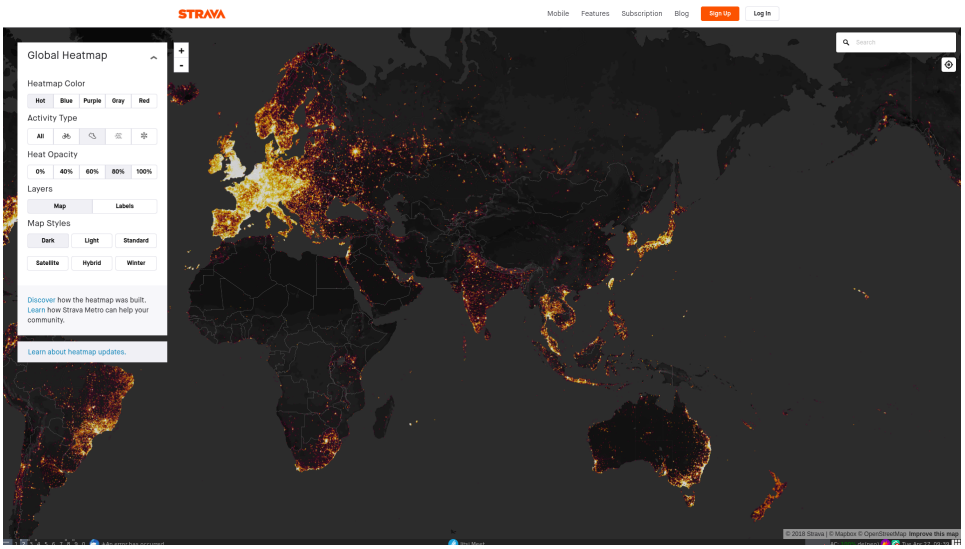
- recon-ng (open source): database aggregation using plenty of modules for retrieval of information from technical sources, interface similar to metasploit.
- shodan.io (freemercial): the database for existing vulnerabilities, should be the daily read of any admin running outfacing systems (but also internal), if you show up on shodan you are essential a sitting duck with a crosshair hovering on you.
- maltego (?): relation tracking

11.16.1 Recon-NG

- Web-Reconnaissance
- Graph of:
  - Hosts, Companies, Domains, Contacts, Vulnerabilities
- Incremental Recon

11.17 Examples

11.17.1 Strava 2018


STRAVA
Mobile
Features
Subscription
Blog
Sign Up
Log In
Search
Global Heatmap
Heatmap Color
Hot
Blue
Purple
Gray
Red
Activity Type
All
Heat Opacity
0%
40%
60%
80%
100%
Layers
Map
Labels
Map Styles
Dark
Light
Standard
Satellite
Hybrid
Winter
Discover how the heatmap was built.
Learn how Strava Metro can help your community.
Learn about heatmap updates.
© 2018 Strava | © Mapbox © OpenStreetMap Improve this map

osint countermeasures

11.18 OSINT Countermeasures

The crucial point here is to not (as in never) publish information you would not share with an adversary. To achieve this, not only do you have to think before you post anything online, but also be aware of the information that already is online.

11.19 OSINT Exercise

Ex. 17 — Become Self-Aware and create a list of all the public places on the Internet where you voluntarily published (or are still publishing) information about yourself and your activities. Include all places that (you think) cannot be linked to yourself, because you are using a pseudonym.

Ex. 18 — Make a list of all the pseudonyms (or names) you have used on these public places. Now draw a graph where you link all pseudonyms that have been mentioned (in any interaction on this page) together. Annotate the links (maybe with a colour) where the interaction links both pseudonyms to the same identity (that is "you").

Ex. 19 — Add your pseudonyms and pages in a fresh recon-ng workspace and try different modules to find out more about yourself. Are you reaching a point, where you are surprised?

Work through CAPEC, OWASP,. . .

- Fuzzing
- Reverse Engineering
- Educated Guessing
- Session Pinning
- Code Injection
- . . .

See Section 11.19.2.

There is a cornucopia of dedicated pentesting hardware to be bought from more-or-less shady looking online stores. Also you will find amble advice on how to build your own. Generally, you probably will not succeed if you don't know your tools and techniques — and no two assignments are ever exactly the same[9]. Thus it seems to be advicable to, at least, extensively customize your tools but also be prepared to code large portions for yourself.

For the most part you will fare well with a box running a Kali-Linux installation. But be aware that the most powerful and flexible tool is your own code and that many "normal" tools can and should be used.

Most tools fall into one of the three categories:

Outdated Exploits might open up the target at the blink of an eye, but actually could be executed by a monkey with a typewriter and show only the existence of bugs that should have been squashed a long time since. That said, you still get your vulnerability report, but the fame of discovery belongs to the authors of the tool. Nonetheless, a collection of vulnerability scanners should be part of every testers toolkit, they help to quickly get past the obvious vulnerabilities.

Supportive Tools can make your life much more enjoyable because they can automate a lot of the grinding repetitive work. Fuzzers and repeaters,

[9] at least the interesting ones

as well as generators and encoders for payloads belong in this category. They are very helpful, but you have to know how to use them — of course. This has to be practised.

Media Access provide hard- or software for specific communication channels. Software-defined Radios, Programmable USB-Sticks, RFID Readers, and all types of cables and adaptors fall in this category. Go well prepared into your assignment, there is nothing worse than standing in front of a server rack and missing the standard key, or bringing an x 802.11 to an RJ45 battle.

11.19.1 Guidewords

Analysis by Guidewords is a method to

STRIDE

For the analysis of threats and vulnerabilities within the Security Development Lifecycle (SDL) STRIDE[10] analysis aims at covering all possible angles of attack by working through the enumeration of potential weaknesses:

S poofing Identity,

T ampering with Data,

R epudiation,

I nformation Disclosure,

D enial of Service and

E levation of Privilege

For the analysis the system is compartmentalised and each component is analysed for each STRIDE vulnerability.

11.19.2 Schwachstellenbewertung

DREAD

The DREAD risk assessment model has been developed by Microsoft. It provides a separation of different topics that, individually, could be assessed to generate an overall threat level for individual threats.

[10] `http://msdn.microsoft.com/en-us/magazine/cc163519.aspx`

STRIDE	Example Attack
Spoofing	Cookie Replay
	Session Highjacking
	CSRF
Tampering	XSS
	SQL Injection
Repudiation	Audit Log Deletion
	Insecure Backup
Information Disclosure	Eavesdropping
	Verbose Exception
Denial of Service	Website defacement
Elevation of Privilege	Logic Flow Attacks

Table 11.4: Common Attacks related to STRIDE [https://www.owasp.org/images/a/a6/AdvancedThreatModeling.pdf]

D amage - how bad would an attack be?
R eproducibility - how easy is it to reproduce the attack?
E xploitability - how much work is it to launch the attack?
A ffected users - how many people will be impacted?
D iscoverability - how easy is it to discover the threat?

Select High (3), Medium (2), or Low (1)

The severity of individual threats is evaluated by determining a risk level for each individual DREAD category (Table 11.5) and summarising them as provided by example in Table 11.6.

The severity level of every threat depends on the

threat ranges: 12-15 High, 8-11 Medium, 5-7 Low

The threat evaluation is summarised in a *Threat Description Table* (Table 11.7) containing rows for Threat Description, Threat target, Risk rating, Attack techniques and Proposed countermeasures.

In practice the procedures and reports are adapted to the needs of security analyst and even scenario.

Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System (CVSS) provides a method to estimate and communicate the severity of security vulnerabilities. [**cvss31**] There is a very throughout introduction into the use of CVSS at the First webpage[11]

[11] https://www.first.org/cvss

Rating	High (3)	Medium (2)	Low (1)
Damage Potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.	Leaking sensitive information	Leaking trivial information
Reproducability	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced, but only with a timing window and a particular race situation.	The attack is very difficult to reproduce, even with knowledge of the security hole.
Exploitability	A novice programmer could make the attack in a short time.	A skilled programmer could make the attack, then repeat the steps.	The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
Affected users	All users, default configuration, key customers	Some users, non-default configuration	Very small percentage of users, obscure feature; affects anonymous users
Discoverability	Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.	The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use.	The bug is obscure, and it is unlikely that users will work out damage potential.

Table 11.5: DREAD Threat Rating Table [`http://msdn.microsoft.com/en-us/library/aa302419.aspx#c03618429_011`]

Threat	D	R	E	A	D	Total	Rating
Attacker obtains authentication credentials by monitoring the network.	3	3	2	2	2	12	High
SQL commands injected into application.	3	3	3	3	2	14	High

Table 11.6: Example DREAD Threat Level Evaluation

Threat Description	Attacker obtains authentication credentials by monitoring the network
Threat target	Web application user authentication process
Risk rating	High
Attack techniques	Use of network monitoring software
Countermeasures	Use SSL to provide encrypted channel

Table 11.7: Threat Summary Table

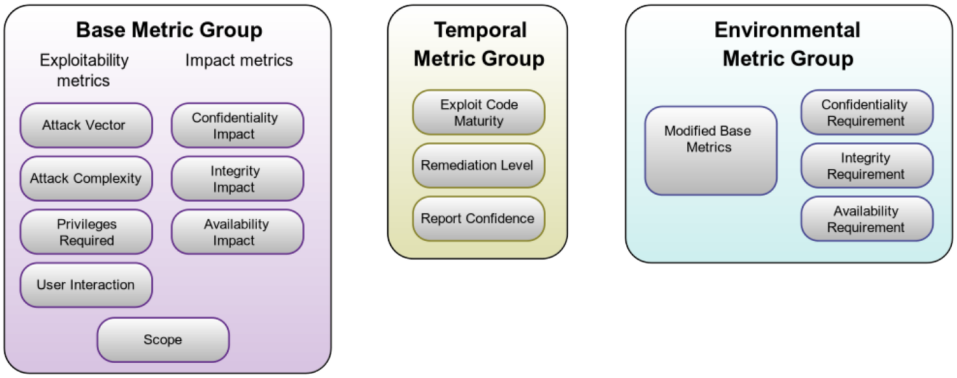
- Metric for Severity
- Communication on Vulnerabilities
- Base Score
- Temporal Score allows to assess temporary variable factors, i. e., maturity of available exploit implementations
- Environmental Metric Group for quantification of environmental security requirements and and modified base metrics

Severity Score $[0, \ldots, 10]$

Figure 11.38: Example CVSS Value.

Rating	Score
None	0
Low	0.1 - 1.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Table 11.8: CVSS Severity Scores



Description of the non-obvious scores:

Scope Can the vulnerability affect resources outside the vulnerable scope? For example, can the vulnerability of the kernel affect a database deployed at the system (most likely “yes”) or can the vulnerability in one user application affect the application of a different user (hopefully not without further vulnerabilities).

`Base Score: 7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`

Attack Vector: Network (0.85)

Attack Complexity: High (0.44)

Privileges Required: None (0.85)

User Interaction: None (0.85)

Scope: Unchanged (special)

Confidentiality Impact: High (0.56)

Integrity Impact: None (0.00)

Availability Impact: None (0.00)

$$
\begin{aligned}
ISS &= 1 - [(1 - C)(1 - I)(1 - A)] \\
&= 1 - [(1 - 0.85) \cdot 0 \cdot 0] = 0.85 \\
Impact &= 6.42 \times ISS : S = Unchanged \\
&= 0.85 = 5.457 \\
Exploitability &= 8.22 \times AV \times AC \times PR \times UI \\
&= \times 0.85 \times 0.44 \times 0.85 \times 0.85 = 2.22117 \\
BaseScore &= [Min[(Impact + Exploitability 5.457 + 2.22117), 10]] \\
&= 7.6 \text{High Severity}
\end{aligned}
$$

12

IT-Security Managment

Abbildung 12.10 illustriert den Zusammenhang zwischen Angreifern (Threat Agents) und Eigentümern (Owners) von Sachen (Assets). Eigentümern eines Assets Zeichnen sich insbesondere dadurch aus, dass sie, innerhalb eines gegebenen sozialen und rechtlichen Kontext, über die Verwendung eines Asssets Entscheiden dürfen. Das Verhältnis zwischen Angreifern und Eigentümern zeichnet sich wiederum dadurch aus, das sie unterschiedliche Ziele bezüglich der Geheimhaltung, Integrität oder Verfügbarkeit der betreffenden Sachen haben.

Angreifer stellen eine Bedrohung (Threat) dar, wenn sie ihre Ziele mit oder an der Sache, ohne die nötigen Rechte zur Durchführung[1], durchsetzen, oder planen durchzusetzen. Bedrohungen können sich wiederum nur durch die Ausnutzung von Schwachstellen (Vulnerabilities) manifestieren. Die Möglichkeit dass eine Bedrohung sich an einer Sache manifestiert wird als Risiko (Risk) bezeichnet. Um dies zu verhindern, können Eigentümer Maßnahmen (Countermeasures) gegen Schwachstellen ergreifen.

Ein einfaches Beispiel, welches vermutlich jedem aus dem eigenen Leben vorstellbar ist, ist der Einbau eines neuen Schließzylinders in die Wohnungstür um der Bedrohung durch einen neugierigen Vermieter zu begegnen. Dadurch kann die Schwachstelle eines vermuteten Zweitschlüssels im Besitz des neugierigen Vermieters geschlossen werden und das Risiko, dass der neugierige Vermieter sich unbefugt Zugang (das Asset) zur eigenen Wohnung verschafft reduziert werden. Es mag sein, dass die Rechtslage vorsieht, dass der Vermieter ein Recht auf den freien Zugang zur Wohnung hat, dann hätten wir mit unserem Zylindertausch einen Angriff auf dieses Recht durchgeführt.

Interessanterweise kann es sogar so sein, dass dieselbe Handlung in unterschiedlichen Kontexten erlaubt oder verboten ist. Und natürlich ist deshalb auch

[1] innerhalb eines verbreitet akzeptierten Kontext von rechtlichen, sozialen und ethischen Richtlinien

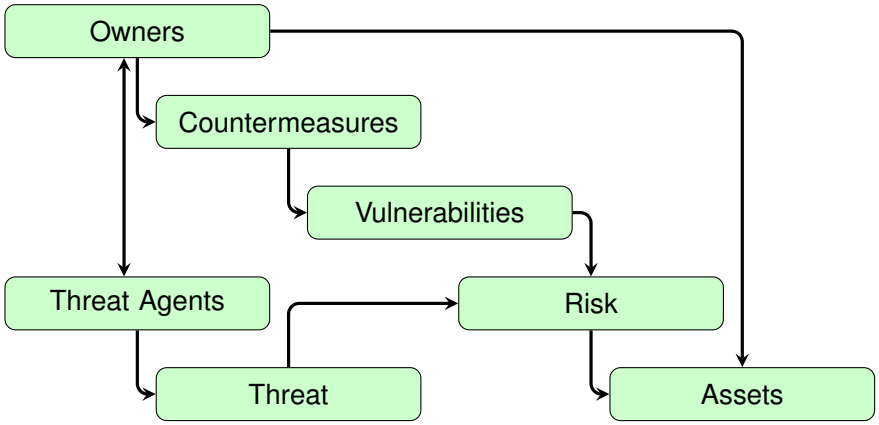


Figure 12.1: Security concepts and relationships [**stallings2008computer**][**CommonCriteria**]

die Unterscheidung zwischen Angriffswerkzeugen und Werkzeugen nicht immer eindeutig möglich.

12.1 Rechtlicher & staatlicher Rahmen

- NIS Directive [**NIS-Directive-2016**] Europäische Gesetznorm zur Einführung von nationaler Cyber-Sicherheits-Gesetzgebung
- Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme (IT-Sicherheitsgesetz – ITSiG, 2015, Artikelgesetz), beeinflusst (uA)
  - Telekommunikationsgesetz (TKG)
  - Telemediengesetz (TMG)
  - Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSI-Gesetz – BSIG) [**BSIG2015**]
  - Gesetz über die Elektrizitäts- und Gasversorgung (Energiewirtschaftsgesetz – EnWG)
  - ...
- IT-Sicherheitskatalog gemäß § 11 Absatz 1a Energiewirtschaftsgesetz
- Verordnung zur Bestimmung Kritischer Infrastrukturen nach dem BSI-Gesetz (BSI-Kritisverordnung – BSI-KritisV) [**BSI-KritisV2016**, **BSI-KritisV2017**]
- Artikelgesetz
- Telemedien/Telekommunikation nach “Stand der Technik” sichern
- Vorsatz oder fahrlässig:

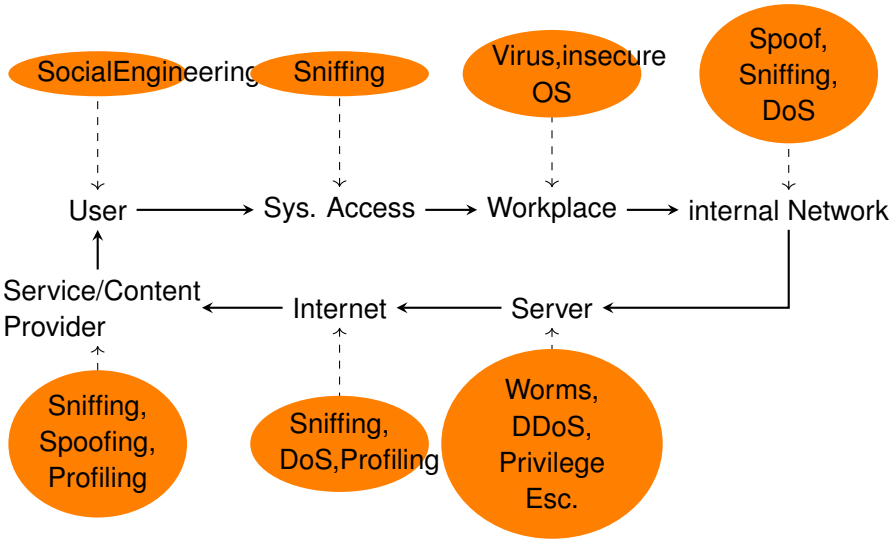


Figure 12.2: Chain of system security-relevant components. If the security of a single one is compromised, the whole system is jeopardised.[Eckert 2011, p.40]

bibtex reference needed

  - Bußgeld bis 50k€ (KRITIS: 100k€)

  - ggF. persönliche Haftung

- BSiG § 8c: Sicherheitsanforderungen digitale Dienste

  - Maßnahmen: Sicherheit, Kontinuität, Erkennung,...

  - Meldepflicht von Vorfällen

Gesetz
zur Erhöhung der Sicherheit informationstechnischer Systeme
(IT-Sicherheitsgesetz)*

Vom 17. Juli 2015

Der Bundestag hat das folgende Gesetz beschlossen:

Artikel 1

Änderung des
BSI-Gesetzes

Das BSI-Gesetz vom 14. August 2009 (BGBl. I S. 2821), das zuletzt durch Artikel 3 Absatz 7 des Gesetzes vom 7. August 2013 (BGBl. I S. 3154) geändert worden ist, wird wie folgt geändert:

1. § 1 wird wie folgt gefasst:

„§ 1

Bundesamt für
Sicherheit in der Informationstechnik

Der Bund unterhält ein Bundesamt für Sicherheit in der Informationstechnik (Bundesamt) als Bundesoberbehörde. Das Bundesamt ist zuständig für die Informationssicherheit auf nationaler Ebene. Es untersteht dem Bundesministerium des Innern."

2. Dem § 2 wird folgender Absatz 10 angefügt:

„(10) Kritische Infrastrukturen im Sinne dieses Gesetzes sind Einrichtungen, Anlagen oder Teile davon, die

1. den Sektoren Energie, Informationstechnik und Telekommunikation, Transport und Verkehr, Gesundheit, Wasser, Ernährung sowie Finanz- und Versicherungswesen angehören und
2. von hoher Bedeutung für das Funktionieren des Gemeinwesens sind, weil durch ihren Ausfall oder ihre Beeinträchtigung erhebliche Versorgungsengpässe oder Gefährdungen für die öffentliche Sicherheit eintreten würden.

Die Kritischen Infrastrukturen im Sinne dieses Gesetzes werden durch die Rechtsverordnung nach § 10 Absatz 1 näher bestimmt."

3. § 3 wird wie folgt geändert:
   a) Absatz 1 Satz 2 wird wie folgt geändert:
      aa) In Nummer 2 werden die Wörter „zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" durch die Wörter „erforderlich ist, sowie für Dritte, soweit dies zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" ersetzt.
      bb) In Nummer 15 werden die Wörter „kritischen Informationsinfrastrukturen" durch die Wörter „Sicherheit in der Informationstechnik Kritischer Infrastrukturen" und der Punkt am Ende durch ein Semikolon ersetzt.
      cc) Die folgenden Nummern 16 und 17 werden angefügt:
         „16. Aufgaben als zentrale Stelle im Bereich der Sicherheit in der Informationstechnik im Hinblick auf die Zusammenarbeit mit den zuständigen Stellen im Ausland, unbeschadet besonderer Zuständigkeiten anderer Stellen;
         17. Aufgaben nach den §§ 8a und 8b als zentrale Stelle für die Sicherheit in der Informationstechnik Kritischer Infrastrukturen."
   b) Folgender Absatz 3 wird angefügt:
      „(3) Das Bundesamt kann Betreiber Kritischer Infrastrukturen auf deren Ersuchen bei der Sicherung ihrer Informationstechnik beraten und unterstützen oder auf qualifizierte Sicherheitsdienstleister verweisen."
4. Die Überschrift von § 4 wird wie folgt gefasst:

* Notifiziert gemäß der Richtlinie 98/34/EG des Europäischen Parlaments und des Rates vom 22. Juni 1998 über ein Informationsverfahren auf dem Gebiet der Normen und technischen Vorschriften und der Vorschriften für die Dienste der Informationsgesellschaft (ABl. L 204 vom 21.07.1998, S. 37), zuletzt geändert durch Artikel 26 Absatz 2 der Verordnung (EU) Nr. 1025/2012 des Europäischen Parlaments und des Rates vom 25. Oktober 2012 (ABl. L 316 vom 14.11.2012, S. 12).

Das Bundesgesetzblatt im Internet: www.bundesgesetzblatt.de | Ein Service des Bundesanzeiger Verlag www.bundesanzeiger-verlag.de

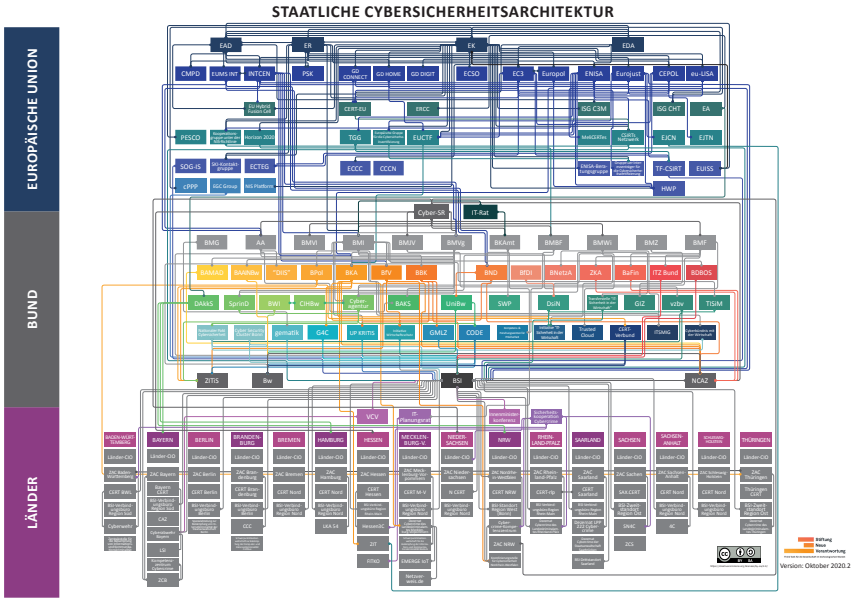
- BSI Gesetz
  - Einführung ISBÖFI (Infrastruktur im besonderem öffentlichen Interesse)
    - e. g., Rüstungsindustrie
  - Anzeigepflicht beim Einsatz kritischer Komponenten.Nach §2 Abs. 13 solche Komponenten die wesentlich für die Erbringung der Dienste in kritischen Infrastrukturen sind. In einer strengen Auslegung können das sehr viele Komponenten sein. Das BMI hat Erlaubnisvorbehalt.
  - Geheimhaltung von Schwachstellen, wobei das BSI hier auf Weisung des BMI handeln müsste. Zerstört nach Ansicht vieler Experten das Vertrauensverhältnis von White-Hat Sicherheitsforschern und BSI.

- ...

Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSiG)

- §§ 4 und 8b: Zentrale Meldestelle (Bund, KRITIS)
- § 5: Gefahrenabwehr für Bund TK
- § 5a: Incidence Response (Bund, KRITIS) in herausgehobenen Fällen
- § 7a: Produktuntersuchung
- § 8c: Sicherheits und Meldepflicht
- § 9: Zertifizierung, Personen und Systeme
- § 14: Bußgelder 50k/100k€

12.1.1 Behördenstrukturen Cyberabwehr in Deutschland



- BSI
  - Nationales Cyber-Abwehrzentrum (Cyber-AZ)
  - Allianz für Cybersicherheit
  - Nationalen IT-Lagezentrum/IT-Krisenreaktionszentrum
- UP-KRITIS (BSI und BKK)
- Zentrale Stelle für Informationstechnik im Sicherheitsbereich (ZITIS)
- Agentur für Cybersicherheit `https://www.bundesregierung.de/breg-de/themen/digital-made-in-de/agentur-fuer-innovation-in-der-`
- Bundeswehr
  - Kommando Cyber- und Informationsraum (CIR)
- Bundeskriminalamt/Landeskriminalämter
- Bundesnachrichtendienst
- Bundes-/Landesverfassungsschutz

12.2 Tasks and Duties of Managment

- Assumption of overall responsibility for...

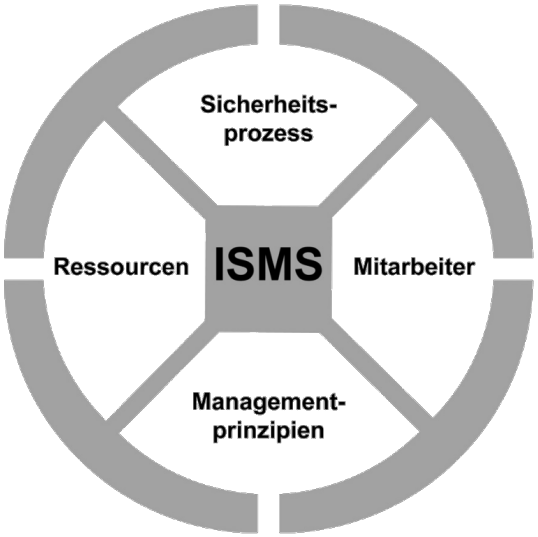


Figure 12.3: Components of an ISMS

- Integrating...
- Managing and maintaining...
- Setting achievable goals
- Weighing up security costs against benefits
- Be role model

This list is defined in [**bsi08InformationSecurityManagment**]

> "An information security management system [...] specifies the instruments and methods that the administration/management level of an institution should use to comprehensibly manage the tasks and activities aimed at achieving information security."
> [**bsi08InformationSecurityManagment**]

- **instruments and methods** of the administration/management level
- to **manage**
- tasks and activities
- to achieve **information security**

[**bsi08InformationSecurityManagment**]

security levels

Security levels are determined with respect to the consequences of a security breach. *Very high* levels are used if failure has existential consequences for an organisation or large parts of society or economy. *High* levels are used if

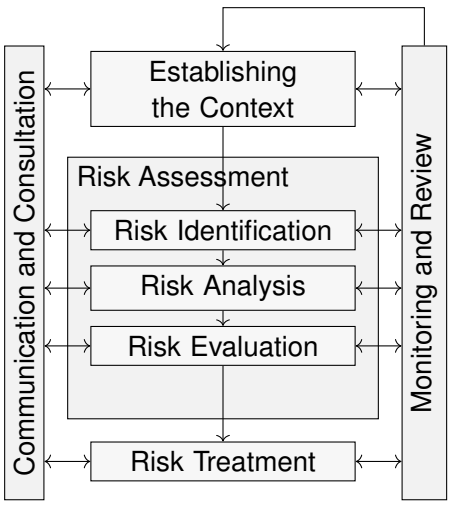


Figure 12.4: Risikomanagementprozess nach ISO/IEC 31000

VERY HIGH	Existential Consequences for Organisation or Large Parts of Society
HIGH	Non-Functional Large Areas, Significant Impairment, Affects Third Parties
NORMAL	Impairment of Processes
none	No significant consequences

Table 12.1: Threat Levels

damages render key areas of an organisation non-functional, the damage leads to significant impairment of the organisation or affects third parties. *Normal* levels are used if any level of impairment is to be expected on security fails. If no impairment is to be expected, no security level is assigned, i. e., the component/objective is of no importance to security. (Table 12.1)

12.2.1 Sicherheitskataloge und -empfehlungen

Die folgende Auflistung soll einen ersten Überblick über die umfangreiche Landschaft von Schutzkatalogen und Bewertungsmethoden in kritischen Infrastrukturen, aus der Sicht der Stromversorgung geben. Die Kenntnis zumindest dieser Kataloge ist essentiell für die Arbeit in der IT-Sicherheit in Europa, da sie essentiell das Sprachgerüst für die Organisation von Sicherheitsmaßnahmen darstellen.

In der technischen Praxis sind diese Kataloge hilfreich bei der Identifizierung von Schutzlücken und Maßnahmen. Eine wesentliche Funktion ist aber letztlich, dass die Sicherheit einer Organisation einem strukturierten Vorgehen folgt. Diese Vorgehen wird oft unter dem Stichwort Information Security Man-

agement System (ISMS) geführt, was in Gänze aber in separaten Vorlesungen behandelt werden muss.

Europa/International:

- ENISA
  - Railway Cybersecurity
  - Port Cybersecurity - Good practices for cybersecurity in the maritime sector
  - National Electric Sector Cybersecurity Organization Resource [**Marinos2013**]
- SGIS Toolbox [**M490**] Ausgehend von Use-Cases und einer SGAM-Modellierung werden Anforderungen, Schutzlücken und Maßnahmen aus SGIS-Katalogen abgebildet. Die SGI
- BDEW Whitepaper [**BDEW˙White˙2018**]
- Empfehlungen des National Cyber Security Center (NCSC)
- BSI-Standards [**bsi200-1**, **bsi200-2**, **bsi200-3**]
- UP KRITIS Best-Practice [**upkritis2017bestpractice**]
- BNetzA
  - IT-Sicherheitskatalog für Strom- und Gasnetze
  - IT-Sicherheitskatalog für Betreiber von Energieanlagen
- ISO/IEC 27k-Standards
  - 27001: ISMS-Specification
  - 27032, 27033, 27034: cyber-, network, application security
  - ISO/IEC TR 27019: Information security management guidelines based on ISO/IEC 27002 for process control systems specific to the energy industry
- IEC 62443 Industrial communication networks - Network and system security

USA:

- NIST Framework for Critical Infrastructures [**NIST2017Framework**] liefert grundlegende Terminologie und Klassifizierung von Maßnahmen, die — leider — nicht mit den Maßnahmenkatalogen in [**NISTIR7628**] übereinstimmt.
- NIST SP 800 Series

- NISTIR 7628 [**NISTIR7628**] Auswahl eines Maßnahmenkatalogs basierend auf modellierten Systemstruktur und Schutzklassen für unterschiedliche Schnittstellen. Anhand der Kritikalität werden Maßnahmen aus einem Maßnahmenkatalog ausgewählt. Bietet eine vollständige Methodik zum Grundschutz, sowie eine Anleitung zum vorgehen für besonders kritische Bereiche.
- NERC Critical Infrastructure Protection (CIP) Standards
- NESCOR Szenarienmethode und -katalogeNutzung der Szenarienmethodik zur Identifizierung von Schutzlücken durch Wargame-Sessions

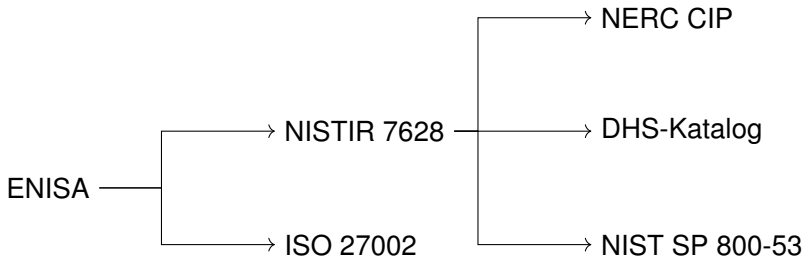


Figure 12.5: Hierarchische Abbildbarkeit von Maßnahmenkatalogen in den unterschiedlichen Richtlinien und Handbüchern. (Oder: wer könnte bei wem abgeschrieben haben?)

Und auch in der Rechtsprechung wird das Thema Sicherheit mittlerweile prominent verhandelt:

12.3 IT-Security Process

> Security is a process, not a product.
>
> Products provide some protection, but the only way to effectively do business in an insecure world is to put processes in place that recognize the inherent insecurity in the products. The trick is to reduce your risk of exposure regardless of the products or patches.
>
> Bruce Schneier:
> Computer Security: Will We Ever Learn? [seen 2020-08-26]

The Deming-Cycle (Figure 12.6) is a general concept for management. It establishes that any management process begins with a plan, that is then executed. After execution good management evaluates the results and decides on actions based on this evaluation. Not only IT-security management comprises a never ending cycle, but in IT-security at least, this has been made explicit by Bruce Schneier:

This is valuable to keep in mind if one intends to create secure products.

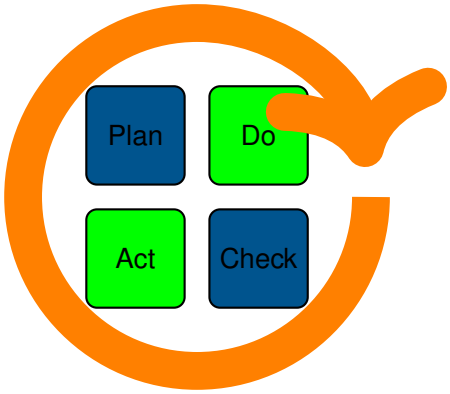
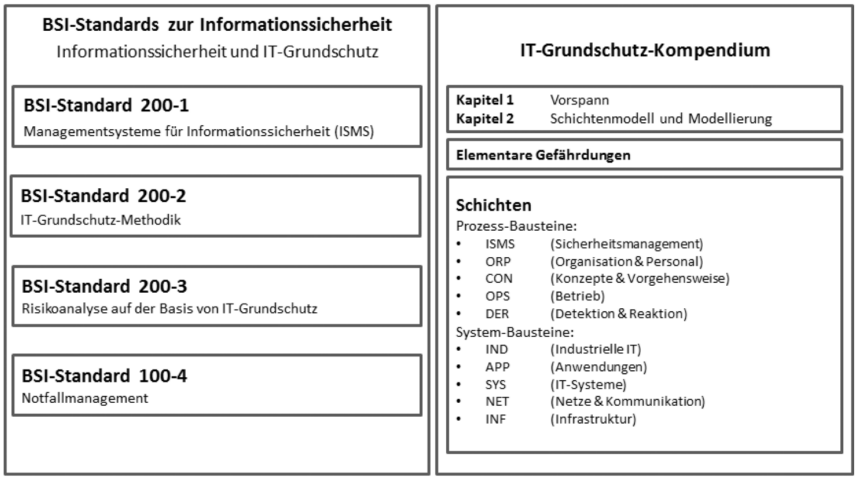


Figure 12.6: PDCA Managment Cycle

The “Bundesamt für Sicherheit in der Informationstechnik” (BSI) has defined a standard procedure for the establishment of an Informatios Security Managment System. The Standard 100 contains four separate works considering the topics *managment*, *Grundschutz* (baseline security), *Risk Analysis*, and *Emergency Managment*.

- 200-1: Managementsysteme für Informationssicherheit (ISMS)
  - Kompatibel zu ISO 27001
- 200-2: IT Grundschutz Methodik
  - Konkrete Handlungsanleitung für Basis-, Kern-, und Standard-Schutzlevel.
  - Baukastenprinzip für
  - Anleitung am Beispiel
- 200-3: Risikomanagement
- 100-4: Notfallmanagement



[BSI 200-1, Abb. 1, `https://www.bsi.bund.de`]

Based on BSI Standard 100-1 and 100-2 we summarise the IT-Security Process. Please refer to [5] for an introduction in german language.

0. Initiation
1. Security Concept
2. Implementation
3. Maintenance and Improvement

Initiation

1. Accepting Responsibility (Mgmt)
2. Design and Plan of the Security Process
3. Creation of Security Policy
4. Organisation of Security Process
5. Providing Resources
6. Integration of all Employees

Accepting Responsibility by the management:

1. Managment is informed about risks and consequences
2. Managment assumes responsibility
3. Managment initiates process within organisation

Design and plan of the security process:

1. Appoint contact persons for all business processes and specialised tasks

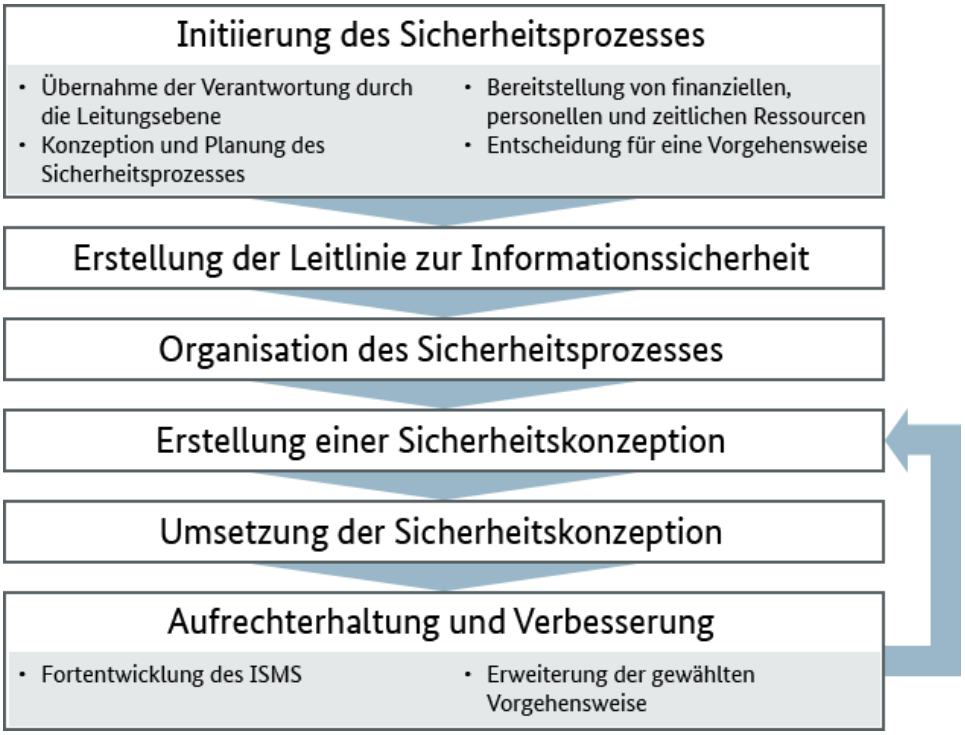


Figure 12.7: Phasen des Sicherheitsprozesses nach BSI 200, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzSchulung/OnlinekursITGrundschutz2018/Lektion_2_Sicherheitsmanagement/Lektion_2_02/Lektion_2_02_node.html;jsessionid=6FCE91C1D11E131A548B603DB8C89776.1_cid360

2. Perform a rough assessment of the value of the information, business processes, and specialised tasks

3. Determine the general requirements

4. Estimate the importance of the business processes, specialised tasks, and information

5. Specify the general information security objectives

6. Obtain the agreement of management

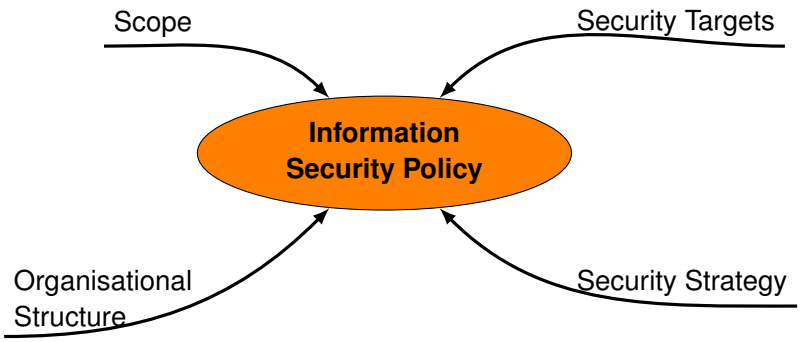
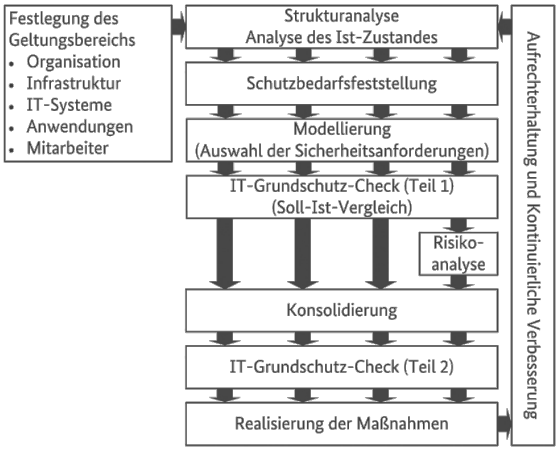


Figure 12.8: Contents of a Security Policy



12.3.1 Security Policy

A security policy must describe in general terms how security is achieved in an organisation. It must describe which assets are to be protected and what kind of resources are utilised in which way for protection. The must describe the objectives and the desired level of protection. (Figure 12.8)

1. Obtain a request from management
2. Specify the scope
3. Summon a development group
4. Organise management approval
5. Release the security policy
6. Check the security policy regularly

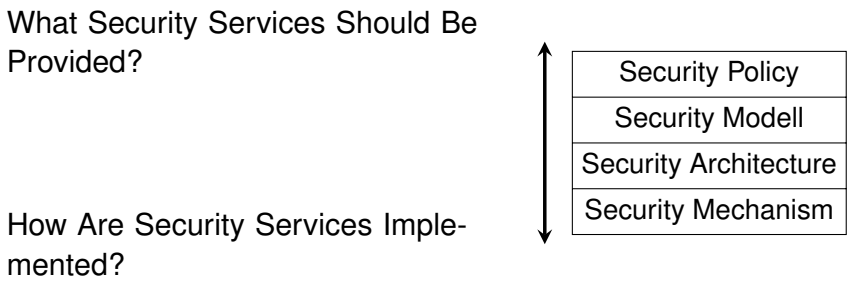


Figure 12.9: Security Strategy Stack

12.3.2 Security Strategy

A security strategy has to provide answers how a system is secured on every layer. The Security Strategy Stack (Figure 12.9) shows the different layers which have to be considered. From top to bottom the layers provide increasingly detailed realisations of the layer above.

Security Policy: Guidelines and Directives, the top level security requirements definition.

Security Model/Concept: Representation of the IT-Security Infrastructure, Connections between Services, Dependencies, etc.

Security Architecture: structural realisation describing how the model is realised in the real world.

Security Methods: which techniques and services shall be implemented to provide the security requirements as posed by the architecture?

Security Concept

12.4 Sicherheitsgrundfunktionen

[Dieser Abschnitt ist aus [5, pg. 218] entnommen und wird, um Übersetzungsfehler zu vermeiden, in deutscher Sprache präsentiert.]

Die Sicherheitsgrundfunktionen bieten einen Baukasten zur Sicherstellung von grundlegenden Sicherheitseigenschaften und finden im Grundschutz insbesondere dann Anwendung, wenn eine niedrige Schadenshöhe zu erwarten ist. Eine vollständige Bedrohungs- und Risikoanalyse stellt in solchen Fällen eine unnötig hohen Aufwand dar. Grundsätzlich sollte jede Sicherheitsstrategie zumindest Konzepte zur Realisierung der Grundfunktionen enthalten.

Die Sicherheitsgrundfunktionen sind:

- Identifikation und Authentifikation

- Rechteverwaltung
- Rechteprüfung
- Beweissicherung
- Wiederaufbereitung
- Gewährleistung der Funktionalität

[5, pg. 218]

12.4.1 Identifikation und Authentifikation

- Abwehr von Maskierungsangriffen (Spoofing)
- Wann und Wer wird authentifiziert?
  - Welche Aktionen erfordern (welche Form der) Identität
  - z.B.: Systemzugang, DB-Zugriff,. . .
- Fehlerbehandlung

12.4.2 Rechteverwaltung

- Welche Subjekte auf
- Welche Objekte unter
- Welchen Bedingungen, mit
- Welchen Rechten?
- Rechtegranularität

12.4.3 Rechteprüfung

- Policy Enforcement Points
- Bsp.: `open` file vs. `read` file
- Ausnahmebehandlung bei unauthorisierten Zugriffsversuchen

12.4.4 Beweissicherung

- Unabstreitbarkeit/Non-Repudiation herstellen
- Welche Ereignisse Wie protokollieren
- Zugriff auf Protokolle?
- Fälschbarkeit der Protokolle?

12.4.5 Wiederaufbereitung

- z.B. Hauptspeicher, Festplatten, USB-Sticks
- Informationen sicher entfernen

12.4.6 Gewährleistung der Funktion

Unterthema Backup:

> "Nobody is interested in Backups, what everybody wants is Restore." Michi Nagorsnik
>
> "Only wimps use tape backup: real men just upload their important stuff on ftp, and let the rest of the world mirror it ;)" Torvalds, Linus (1996-07-20)
>
> Kein Backup — Kein Mitleid!

Abbildung 12.10 illustriert den Zusammenhang zwischen Angreifern (Threat Agents) und Eigentümern (Owners) von Sachen (Assets). Eigentümern eines Assets Zeichnen sich insbesondere dadurch aus, dass sie, innerhalb eines gegebenen sozialen und rechtlichen Kontext, über die Verwendung eines Asssets Entscheiden dürfen. Das Verhältnis zwischen Angreifern und Eigentümern zeichnet sich wiederum dadurch aus, das sie unterschiedliche Ziele bezüglich der Geheimhaltung, Integrität oder Verfügbarkeit der betreffenden Sachen haben.

Angreifer stellen eine Bedrohung (Threat) dar, wenn sie ihre Ziele mit oder an der Sache, ohne die nötigen Rechte zur Durchführung[2], durchsetzen, oder planen durchzusetzen. Bedrohungen können sich wiederum nur durch die Ausnutzung von Schwachstellen (Vulnerabilities) manifestieren. Die Möglichkeit dass eine Bedrohung sich an einer Sache manifestiert wird als Risiko (Risk) bezeichnet. Um dies zu verhindern, können Eigentümer Maßnahmen (Countermeasures) gegen Schwachstellen ergreifen.

Ein einfaches Beispiel, welches vermutlich jedem aus dem eigenen Leben vorstellbar ist, ist der Einbau eines neuen Schließzylinders in die Wohnungstür um der Bedrohung durch einen neugierigen Vermieter zu begegnen. Dadurch kann die Schwachstelle eines vermuteten Zweitschlüssels im Besitz des neugierigen Vermieters geschlossen werden und das Risiko, dass der neugierige Vermieter sich unbefugt Zugang (das Asset) zur eigenen Wohnung verschafft reduziert werden. Es mag sein, dass die Rechtslage vorsieht, dass der Vermieter ein Recht auf den freien Zugang zur Wohnung hat, dann hätten wir mit unserem Zylindertausch einen Angriff auf dieses Recht durchgeführt.

[2] innerhalb eines verbreitet akzeptierten Kontext von rechtlichen, sozialen und ethischen Richtlinien

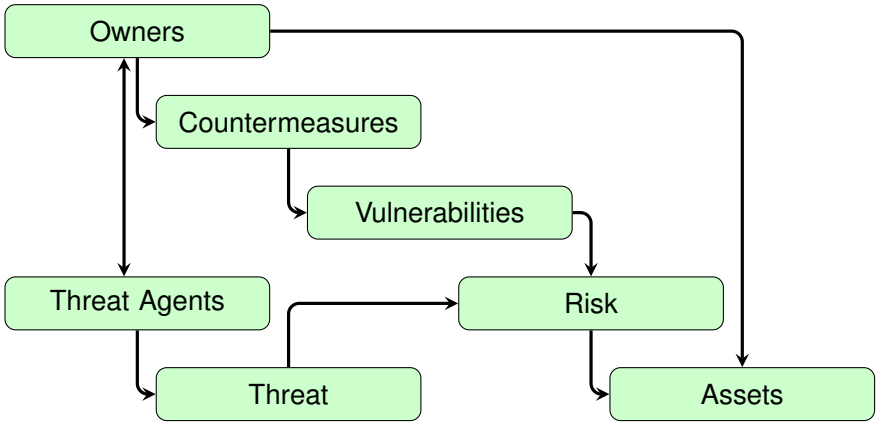


Figure 12.10: Security concepts and relationships [**stallings2008computer**][**CommonCriteria**]

Interessanterweise kann es sogar so sein, dass dieselbe Handlung in unterschiedlichen Kontexten erlaubt oder verboten ist. Und natürlich ist deshalb auch die Unterscheidung zwischen Angriffswerkzeugen und Werkzeugen nicht immer eindeutig möglich.

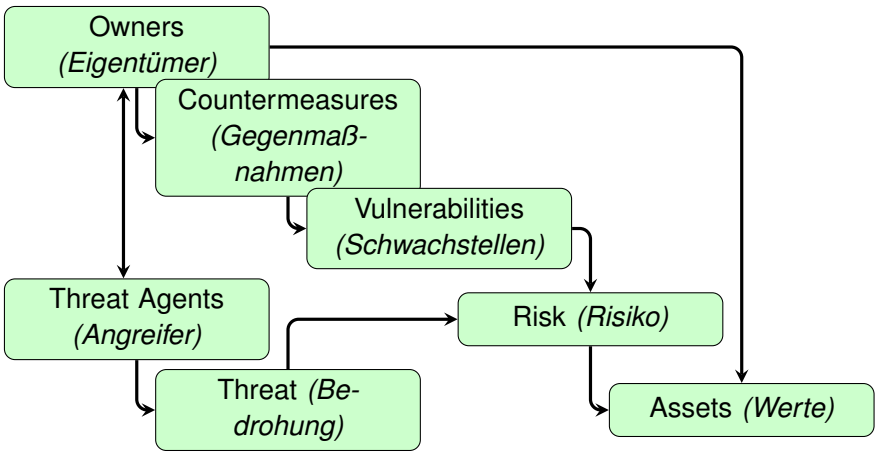


Figure 12.11: Common Criteria: Generisches Angriffsmodell (Übersetzung) [**stallings2008computer**][**CommonCriteria**]

Bibliography

[1] Michael Assante and Robert Lee. “The Industrial Control System Cyber Kill Chain”. In: *SANS Inst. InfoSec Read. Room* (2015).

[2] Jean-Philippe Aumasson. *Serious Cryptography*. no starch press, 2018. URL: https://nostarch.com/seriouscrypto.

[3] Tim Casey. *Understanding Cyberthreat and Motivations to Improve and Defense*. Research rep. 2015.

[4] Timothy (Intel Information Technology Casey. “Threat Agent Library Helps Identify Information Security Risks”. In: *Intel White Pap.* September (2007), p. 12. URL: https://communities.intel.co.jp/servlet/JiveServlet/previewBody/1151-102-1-1111/Threat%20Agent%20Library%7B%5C_%7D07-2202w.pdf.

[5] C. Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. 7th ed. Oldenbourg Wissenschaftsverlag, 2011. ISBN: 9783486706871. URL: http://books.google.de/books?id=x5Y8psQsdnIC.

[6] Barbara Guttman and Edward A Roback. *An introduction to computer security: the NIST handbook*. DIANE Publishing, 1995.

[7] Lockheed Martin. *Gaining the Advantage. Applying Cyber Kill Chain® Methodology to Network Defense*. 2014. URL: http://cyber.%20lockheedmartin.%20com/hubfs/GainingtheAdvantageCyberKill%20Chain.%20pdf.

[8] Bruce Schneier. “Attack Trees”. In: *Dr. Dobbś J. Softw. Tools Prof.* (1999). URL: http://tnlandforms.us/cs594-cns96/attacktrees.pdf%20https://elibrary.ru/item.asp?id=3711965%20http://tnlandforms.us/cns05/attacktrees.pdf.

[9] R. Shirey. *Internet Security Glossary, Version 2*. RFC. IETF, Aug. 2007. URL: https://www.rfc-editor.org/rfc/rfc4949.txt.

[10] Gritta Wolf and Andreas Pfitzmann. “Properties of protection goals and their integration into a user interface”. In: *Computer Networks* 32.6 (2000), pp. 685–700. ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/S1389-1286(00)00029-3. URL: http://www.sciencedirect.com/science/article/pii/S1389128600000293.