Computer System Sicherheit
Notizen WS20/21

Felix Marx

11. Januar 2021

Inhaltsverzeichnis

1 Begriffe

Unterscheidung Betriebssicherheit (safety) und Angriffssicherheit (security).
safety bezeichnet den Schutz vor inneren Fehlern, deren Eintrittswahrscheinlichkeit durch probabilistische Techniken ermittelt wird.
security den Schutz gegen aktive Angreifer.

1.1 Security

Trends und Herausforderungen:
Mobilität, der Übergang zu "smart devices"

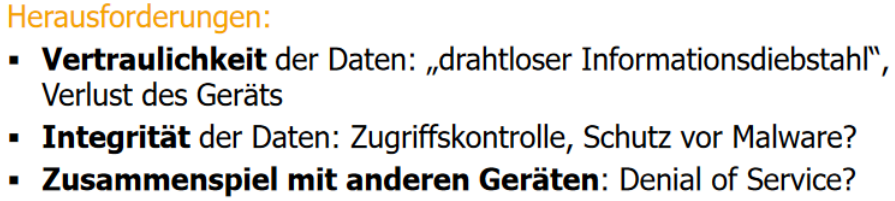


Abbildung 1: Herausforderungen bei der Mobilität

Vernetzung, der Übergang zu "always on" und Automatisierung von Systemen

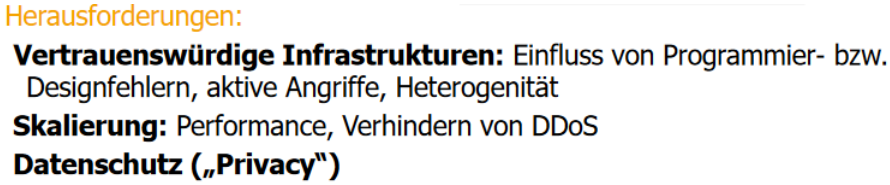


Abbildung 2: Herausforderungen bei der Vernetzung

Miniaturisierung, Produktion kleinerer Chips mit Trend zu Einwegchips

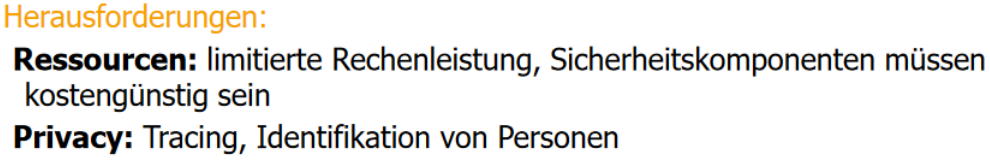
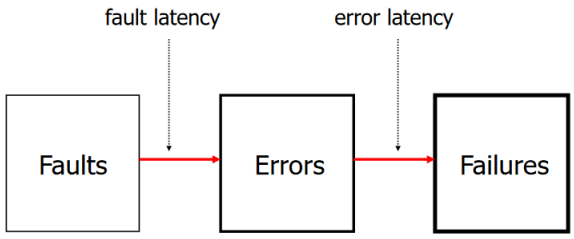


Abbildung 3: Herausforderungen bei der Miniaturisierung

Social Engineering bezeichnet das Erlangen von vertraulichen Daten durch psychologische Techniken
Man-in-the-middle Angriffe, hier werden Nachrichten vor der gesicherten Kommunikation abgefangen.
Lösungen für Phishing umfassen die Verifikation durch Transaktionsnummern (TANs) oder Hardwaretokens was allerding nicht die Integrität der Transaktion gewährleistet.
Schutzmechanismen:
Passive Authentication: Sicherstellung der Authentizität durch Einsatz einer digitalen Signatur
Basic Access Control: elektronische gespeicherte Daten werden erst übermittelt wenn das Lesegerät einen aufgedruckten maschinenlesbaren Code kennt.



- **Fault:** Abnormal condition that can cause an element or an item to fail
- **Error:** Discrepancy between a computed, observed or measured value or condition, and the true, specified, or theoretically correct value or condition.
- **Failure:** Termination of the ability of an element to perform a function as required

Abbildung 4: Ablauf bis ein System versagt

Active Access Control: Verhindert 1:1 Kopien indem ein geheimer Schlüssel in einem gesicherten Chip-Bereich gespeichert wird.
Extended Access Control: Schutz von sensiblen Daten (Fingerabdruck, Iris, etc.) durch das Abgleichen von Zertifikaten die nur eine kurze Zeit gültig sind.

1.2 Safety

Im Fehlerfall sollte ein System immer in den sicheren Zustand übergehen.
Zum Beispiel zeigt die Fahranweisung von Zügen nach oben, sodass es bei einem Schaden am Mechanismus in den Haltezustand nach unten fällt (Schwerkraft). Ein Watchdog (hier Schwerkraft) überwacht quasi das System auf Fehler und resettet es bei einem Fehler.

2 Verlässliche Systeme

Ein zuverlässiges System erfüllt seinen Zweck auch falls Fehler auftreten. Als Beispiel des Ablaufes lässt sich die Einwirkung von kosmischer Strahlung auf eine DRAM-Zelle (Fault) was zum Wechsel eines Wertes führt (Error), wodurch die Berechnung ein falsches Ergebnis liefert (Failure).
In der Zuverlässigkeit wird unterschieden nach:
Availability: Verfügbarkeit eines Systems gemessen in Prozent, d.h.

$$\frac{\text{Total Up Time}}{\text{Total (Up + Down Time)}} = \frac{\text{MTTF}}{\text{MTTF + MTTR}}$$

Reliability: Zuverlässigkeit eines Systems, d.h. die Wahrscheinlichkeit dass ein System über einen gewissen Zeitraum korrekt funktioniert.
Wir bezeichnen mit MTTF die Mean Time To Failure und mit MTTR Mean Time To Recovery. Bei der Berechnung wird für die Up/Downtime nur die Zeiten im vereinbarten Betriebszeitraum gezählt!
Die Wahrscheinlichkeit, dass ein System bis zum Zeitpunkt t fehlerhaft wird lässt sich mittels einer Verteilerfunktion F und der Lebenszeit des Systems T berechnen:

$$F(t) = P(T \leq t),\ R(t) = P(T > t) = 1 - F(t)$$

Die Funktion R hingegen ermittelt die Wahrscheinlichkeit, dass ein System bis zum Zeitpunkt t korrekt funktioniert. Geht man davon aus, dass zukünftige Ausfälle unabhängig davon passieren, wann der letzte Ausfall war, lässt sich die Wahrscheinlichkeit mit einer

Fehlerrate λ so ausdrücken:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$F(t) = \int_{-\infty}^{t} f(x)dx = \qquad \begin{cases} 1 - e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

$$R(t) = 1 - F(t) = \qquad \begin{cases} e^{-\lambda t} & t \geq 0 \\ 1 & t < 0 \end{cases}$$

Mean Time To Failure (MTTF) kann als **Erwartungswert** der Zeit bis zu einem Ausfall bestimmt werden:
Lebenszeit bis Ausfall ist Zufallsvariable $T \sim F$

$$MTTF = E[T] = \int_0^\infty 1 - F(t)\, dt - \overbrace{\int_{-\infty}^0 F(t)\, dt}^{=0} = \int_0^\infty R(t)\, dt$$

Für das exponentielle Modell gilt daher: $MTTF = \int_0^\infty e^{-\lambda t}\, dt = \frac{1}{\lambda}$

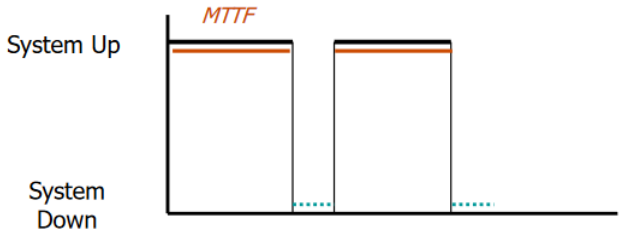


Abbildung 5: Berechnung MTTF mit dem exponentiellen Model

Bei in Reihe geschalteten Komponenten wird die Wahrscheinlichkeit dass das gesamte System korrekt funktioniert R durch $R_{ges}(t) = \prod_{i=1}^{n} R_i(t)$ im exponentiellen Modell wird dann für $\lambda = \lambda_{ges} = \sum_{i=1}^{n} \lambda_i$ verwendet.
Bei Parallelschaltung ist die Wahrscheinlichkeit dass das System korrekt funktioniert:

$$R_{ges}(t) = R_1(t) + R_2(t) - R_1(t) \cdot R_2(t), F_{ges}(t) = F_1(t) \cdot F_2(t)$$

Bei mehr als zwei Komponenten ist die allgemeine Formel:

$$R_{ges}(t) = 1 - \prod_{i=1}^{n}(1 - R_i(t)), F_{ges}(t) = \prod_{i=1}^{n} F_i(t)$$

2.1 Strategien zur Fehlervermeidung/toleranz

- Fehlervermeidung (fault avoidance)
  - Design des Systems stellt sicher, dass Fehler nicht auftreten
  - Beispiel: Testen, Verifikation,...
- Wiederherstellung aus Fehlerzustand (fault recovery)
  - Strategien, um ein System beim Auftreten eines Fehlers wieder in einen korrekten Systemzustand zu bringen

- Fehlertoleranz (fault tolerance)
  - Wenn Fehler nicht vermieden werden können, dann soll das System Fehler tolerieren
  - Beispiele: Redundanz, Safety, ...

Arten von Redundanzen:

- Physikalische Redundanz (physical redundancy)
  - Zusätzliche Ressourcen bzw. Komponenten
  - Berechnungen werden auf mehreren Komponenten ausgeführt und verglichen
  - Statische Redundanz
    - N Systeme laufen parallel
    - $N-1$ Systeme laufen im "stand-by-mode"
    - Bei einem Fehler wird im Betrieb umgeschaltet
    - Dazu muss man den Fehler natürlich zuerst erkennen!
  - Dynamische Redundanz
    - N Systeme laufen parallel
    - Ausfallsicherhere Komponente vergleicht die Resultate
    - Mehrheitsentscheidung (zB. 2-out-of-3)
    - Erkennt Fehler "automatisch"
- Zeitliche Redundanz (temporal redundancy)
  - Berechnungen auf gleicher Hardware-Plattform wiederholen
- Redundanzen durch (Zusatz-)Information (information redundancy)
  - Hinzufügen von zusätzlichen Daten (Checksummen, ...)
  - Erlaubt Datenfelder bei Übertragung oder Speicherung zu erkennen

Bei dynamischer Redundanz wird das Ergebnis parallel berechnet und dann per Mehrheitsentscheidung das korrekte ausgewählt. Problematisch falls im Vergleich ein Problem auftritt. Softwarefehler sind von der Hardware-Redundanz nicht abgedeckt. Zusatzinformationen können die Integrität gesendeter Daten durch Checksummen sicherstellen, allerdings auch nur begrenzt.

Nachteile von Redundanzen sind:

- Schlechtere Performanz (bei temporaler Redundanz)
- Synchronisation erforderlich
- Hohe Kosten durch mehrfache Hardware bzw. mehrfache Implemenation (falls überhaupt möglich)
- Benötigt Mechanismen zur Fehler-Erkennung, welche selbst wieder Fehleranfällig sein können

3 Krypographie

- Vertraulichkeit (Confidentiality)
  - Schutz vor unbefugten Zugriff auf Informationen / Daten
- Integrität
  - Schutz vor Veränderung von Informationen / Daten
- Verfügbarkeit (Availability)
  - Daten oder Systeme sind verfügbar oder erreichbar
- Authentizität (Authenticity)
  - Garantie dass die Daten auch aus der angegebenen Quelle stammen
- Nicht-Abstreitbarkeit (Non-Repudiation)
  - Aktion ist nachprüfbar und kann nicht abgestritten werden

3.1 Begriffe

- Klartext/Nachricht: eine zu verschlüsselnde Information
- Klartextraum: Menge aller möglichen Klartexte
- Chiffrat, Chiffretext: Verschlüsselte Nachricht
- Chiffretextraum: Menge aller möglichen Chiffrate
- Schlüssel: Ein Geheimnis, welches zur Ent-/Verschlüsselung benötigt wird
- Schlüsselraum: Menge aller möglichen Schlüssel
- Verschlüsselung: Umwandlung eines Klartext in ein Chiffrat
- Entschlüsselung: Umwandlung eines Chiffrats in einen Klartext

Kerckhoffs Prinzipien

1. Das System muss praktisch, wenn nicht sogar mathematisch, unentschlüsselbar sein ("Unentschlüsselbarkeit")
2. Es darf keine Geheimhaltung erfordern und darf ohne Schwierigkeiten in die Hände des Feindes fallen ("Keine Geheimhaltung des Systems")
3. Der Schlüssel muss ohne Hilfe geschriebener Notizen kommunizierbar und aufbewahrbar sein und er muss ausgewechselt oder modifiziert werden können nach Belieben der Kommunikationspartner ("Schlüssel ohne Aufschreiben")
4. Es muss anwendbar sein auf die telegraphische Kommunikation
5. Es soll protabel sein und seine Funktion soll nicht die Zusammenkunft mehrerer Personen erfordern
6. Schließlich ist es notwendig, angesichts der Umstände, unter denen es angewendet werden soll, dass das System einfach benutzbar ist und weder große gedankliche Anstrengung erfordert noch die Kenntnis einer langen Liste zu beachtender Regeln ("einfach benutzbar")

3.2 Verschiebechiffre

Die Ceasar-Chiffre ersetzt jeden Buchstaben des Klartextes durch den Buchstaben 3 Stellen weiter rechts, beim Entschlüsseln genau umgekehrt.
Die Verschiebechiffre ersetzt alle Buchstaben durch den Buchstaben welcher eine beliebige aber feste Anzahl an Stellen weiter rechts steht. Der Schlüssel ist die Anzahl an Verschiebungen, d.h. 0,1,2,...,25.

3.3 Mathematische Grundlagen

Als Einwegfunktionen bezeichnet man Funktionen deren Funktionswert $y = f(x) \in Y$ zu einem gegebenen x in polynomieller Zeit berechenbar ist, aber ein Urbild $x \in f^{-1}[\{y\}]$ zu einem gegbenen y nicht in Polynomialzeit bestimmbar ist. Die Existenz von Einwegfunktionen ist nicht bewiesen (P/NP).

3.3.1 Teilbarkeitsregeln

(1) $$a|a$$
(2) $$a|b \wedge b|c \Rightarrow a|c$$
(3) $$a|b \wedge b \neq 0 \Rightarrow |a| \leq |b|$$
(4) $$a|b \wedge b|a \Rightarrow |a| = |b|$$

Sind $a, n \in \mathbb{Z}$ mit $n \neq 0$, dann gibt es eindeutig bestimmte Zahlen $q, r \in \mathbb{Z}$ sodass

$$a = qn + r$$
$$0 \leq r < |n|$$
$$q = \lfloor \frac{a}{n} \rfloor \wedge r = a - qn$$

3.3.2 Größter gemeinsamer Teiler

Wir definieren $\mathbb{Z}_p^\times := \{x \in \mathbb{Z}_p : ggT(x, p) = 1\}$ als die Menge der multiplikativ Inversen in $\mathbb{Z}_p$. Ist p prim, dann ist $(\mathbb{Z}_p^\times, \cdot_p, 1)$ eine zyklische Gruppe.

(5) $$\text{ggT}(a, 0) = |a|$$
(6) $$\text{ggT}(a, 1) = 1$$
(7) $$\text{ggT}(a, b) = \text{ggT}(b, a)$$
(8) $$\text{ggT}(a, b) = \text{ggT}(|a|, |b|)$$
(9) $$\text{ggT}(a, b) = \text{ggT}(b, a - b)$$
(10) $$\text{Für } b \neq 0 \text{ gilt } \text{ggT}(a, b) = \text{ggT}(b, a \bmod b)$$

Lineare diophantische Gleichung: $ax + by = c$
Eine diophantische Gleichung kann wie folgt gelöst werden.
Löse $ax' + by' =$ggT(a,b) und definiere $x = \frac{c}{\text{ggT(a,b)}} \cdot x'$, $y = \frac{c}{\text{ggT(a,b)}} \cdot y'$

Beispiel erweiteter Euklid: $1337 \cdot x + 42 \cdot y = 7$

a	b	$\lfloor \frac{a}{b} \rfloor$	x	y
1337	42	31	-1	$1 - 31 * (-1) = 32$
42	35	1	1	$0 - 1 * 1 = -1$
35	7	5	0	1
7	0		1	0

3.3.3 Primzahlen

Fundamentalsatz der Arithmetik:
Jede natürliche Zahl $n > 1$ besitzt eine Zerlegung in ein Produkt aus Primzahlen, welche bis auf die Reihenfolge eindeutig ist.
Es gibt unendlich viele Primzahlen.

Die Eulersche-Phi-Funktion

$$\varphi(b) = |\{a \in \mathbb{N} : a < b, \text{ggT}(a,b) = 1\}|$$

beschreibt dei Anzahl der zu b teilerfremden Zahlen.
Für eine Primzahl p gilt $\varphi(p) = p - 1$ und $\varphi(p^n) = p^n - p^{n-1}$.
Für teilerfremde Zahlen $a, b \in \mathbb{N}$ gilt $\varphi(ab) = \varphi(a) \cdot \varphi(b)$.
Sind $m, n \in \mathbb{N}$ teilerfremd, dann gilt $m^{\varphi(n)} \text{mod } n = 1$.
Kleiner Satz von Fermat:
Für eine Primzahl p und ein teilerfremde Zahl $m \in \mathbb{N}$ gilt $m^{p-1} \text{mod } p = 1$.

3.4 Symmetrische Kryptosysteme

In symmetrischen Kryptosystemen ist der Schlüssel zum Ver- und Entschlüsseln der selbe. Formal bezeichnet ein symmetrisches Kryptosystem ein 5-Tupel $(\mathcal{M}, \mathcal{K}, \mathcal{C}, e, d)$ mit $\mathcal{M}$ als Menge an Klartexten, $\mathcal{K}$ als Menge von Schlüsseln, $\mathcal{C}$ als Menge an Chiffretexten. $e : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ ist die Verschlüsselungsfunktion und $d : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$ als Entschlüsselungsfunktion. Weiter werden folgende Funktionen eingeführt:

$$\begin{aligned}
num &: \{A, B, C, ..., Z\} \to \{0, 1, 2, ..., 25\} \text{ Zuordnung der Zahlenwerte} \\
char &: \{0, 1, 2, ..., 25\} \to \{A, B, C, ..., Z\} \\
sr &: \{A, ..., Z\} \times \{0, 1, ..., 25\} \to \{A, ..., Z\} \text{ Ist ein Rechtsshift} \\
sl &: \{A, ..., Z\} \times \{0, 1, ..., 25\} \to \{A, ..., Z\} \text{ Linksshift}
\end{aligned}$$

Damit ergibt sich die für Verschiebechiffren die folgenden Interpretationen:

$$\begin{aligned}
e(w_0 w_1 ... w_n, k) &= sr(w_0, k) sr(w_1, k) ... sr(w_n, k) \\
d(w_0 w_1 ... w_n, k) &= sl(w_0, k) sl(w_1, k) ... sl(w_n, k)
\end{aligned}$$

Monoalphabetische Chiffretexte können durch Häufigkeitsanalysen gebrochen werden. Um das zu verhinden existiert das One Time Pad hierbei hat der Schlüssel die selbe Länge wie der Klartext, d.h. jeder Buchstabe im Klartext wird mit einem eigenen Schlüsselbuchstaben verschlüsselt. Das Kryptosystem sieht dann so aus: $(\mathcal{M}^n, \mathcal{K}^n, \mathcal{C}^n, e, d)$
Die Bitweise veroderung ist ein One Time Pad $(\mathbb{Z}_2^n, \mathbb{Z}_2^n, \mathbb{Z}_2^n, e, d)$ mit

$$\begin{aligned}
e &: \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n, e(x, k) = x \oplus k \\
d &: \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n, d(y, k) = y \oplus k
\end{aligned}$$

Ein Kryptosystem heißt perfekt sicher wenn für einen gegebenen Chiffretext jeder Klartext gleich wahrscheinlich ist, d.h. das One Time Pad ist bei gleichverteiltem zufälligen Schlüssel perfekt sicher. Ein Kryptosystem heißt semantisch sicher, wenn es für einen Angreifer der die Länge der Nachricht und das Chiffrat kennt nicht wesentlich einfacher ist auf den Klartext zu schließen als für einen Angreifer der nur die Länge des Klartextes kennt.
Als Blockchiffren werden Kryptosysteme bezeichnet deren Klartexte M^n und Chiffretexte C^n alle eine Blocklänge n und deren Schlüssel K^m die Schlüssellänge m haben. Blockchiffre Verfahren sind unter anderem:

- Advanced Enrcyption Standard (AES) oder Rijndael
  - Blocklänge: $n = 128$
  - Schlüssellänge: $m \in \{128, 192, 256\}$
- Data Encryption Standard (DES)
  - Gebrochen für $n = 64$ und $m = 56$
  - 3DES mit $n = 64$ und $m = 168$ hat die selbe Sicherheit wie 112 Bit Schlüssel
- Serpent ($n = 128$, $m \in \{128, 192, 256\}$)
- Twofish ($n = 128$, $m \in \{128, 192, 256\}$)
- Blowfish ($n = 64$, $32 \leq m \leq 448$ - Standard: $m = 128$)

3.4.1 Vigenère mit alphabetischen Schlüssel, Länge n

Beim Vignerère wird der Klartext entsprechend eines Codewortes verschlüsselt. Dabei wird der aktuelle Buchstabe des Klartextes um die Wertigkeit des aktuellen Buchstabens des Codewortes im Alphabet nach rechts verschoben. Ist man am Ende des Schlüsselwortes angekommen beginnt man wieder am ersten Buchstaben.
Verwendet man einen zufällig erstellten Schlüssel, der genauso lang ist wie der Klartext und benutzt ihn nur ein einziges Mal, so ist die Verschlüsselung perfekt sicher, kann also ohne Schlüssel nicht entschlüsselt werden (One-Time-Pad). Zur Darstellung der Verschiebung kann ein Vigenère-Quadrat verwendet werden. Der Schlüsselraum ist 25^n.

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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

Abbildung 6: Vigenère Quadradt, an der X-Achse ist die Verschiebung und Y-Achse der Klarbuchstabe

3.4.2 Electronic Codebook Modus

Im ECB Modus werden die einzelnen Blöcke im Blockchiffre Kryptosystem jeweils mit dem selben Schlüssel verschlüsselt und die einzelnen Chiffretexte kombiniert um den gesamten Chiffretext zu bekommen. Die Entschlüsselung läuft analog. Da nicht alle Blöcke die genau vom Kryptosystem geforderte Länge besitzen müssen die Blöcke teilweise mit einer Auffüllfunktion $pad : M^* \to (M^n)^*$ gefüllt idealerweise sollte eine pad Funktion wieder umkehrbar sein. Der ECB Modus ist für Klartexte welche in mehrere Blocke augeteilt werden unsicher und sollte nie verwendet werden.
Im ECB Modus ist die Ver- und Entschlüsselung parallelisierbar.

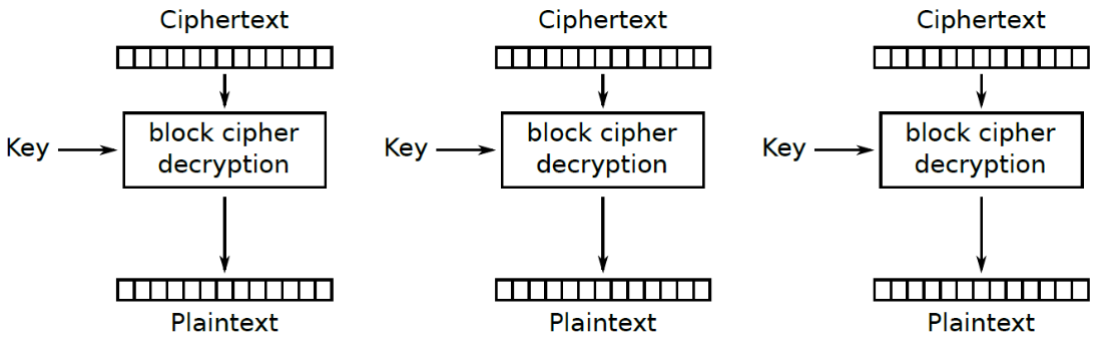


Abbildung 7: Entschlüsselung im ECB Modus

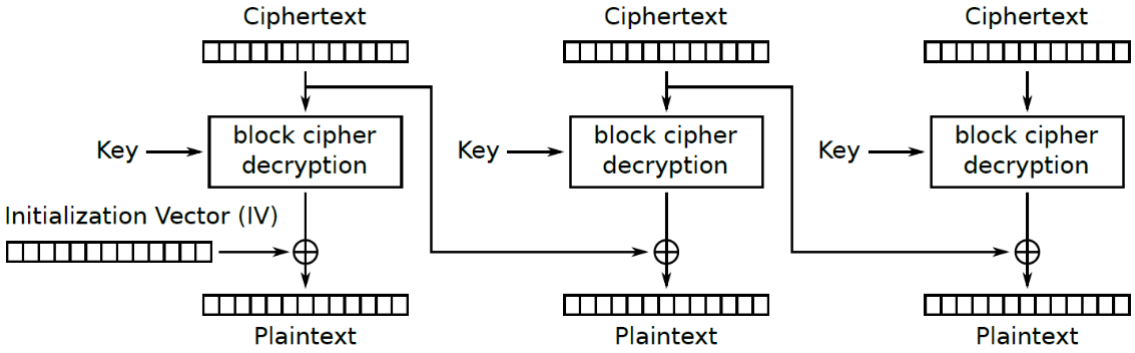


Abbildung 8: Entschlüsselung im CBC Modus, bei der Entschlüsselung umgedreht, sodass Parallelisierung unmöglich ist

3.4.3 Cipher Block Chaining Modus

Der CBC Modus ist eine Verbesserung des ECB Modus, da hier für die Ver- und Entschlüsselung jeweils der vorherige Block als Rauschen mit dem Klartext verodert wird, sodass auch gleiche Muster im Klartext verschiedene Ergebnisse im Chiffretext produzieren. Dafür erweitern wir das Kryptosystem um die Menge der Zufallswerte $\mathcal{R}$ aus welcher der Initialisierungvektor gewählt wird. Hierbei ist $\mathcal{R}$ nicht geheim und kann unverschlüsselt übertragen werden.
Es soll $e(x,k,r_1) \neq e(x,k,r_2)$ gelten. Für die Sicherheit der Verschlüsselung ist wichtig, dass jeder Zufallswert nur einmal verwendet wird.
Das erweiterte Kryptosystem heißt randomisiertes symmetrisches Kryptosystem und wird als 6-Tupel dargestellt: $(\mathcal{M},\mathcal{K},\mathcal{C},\mathcal{C},\mathcal{R},e^*,d^*)$. Der Definitionsbereich von e^* und d^* wird um $\mathcal{R}$ erweitert.

$$y_i = e(x_i \oplus y_{i-1}, k), x_i = d(y_i,k) \oplus y_{i-1} \text{ für } i \geq 1$$

Die Verschlüsselung ist damit nicht mehr parallelisierbar, aber die Entschlüsselung bleibt es.

Damit die Verschlüsselung auch parallelisierbar ist gibt es den Counter (CTR) Modus hier wird bei der Verschlüsselung für jeden Block der Initialiserungsvektor mit einem Zählerwert verschlüsselt und erst dann mit dem Klartext verodert. Die Entschlüsselung läuft genau gleich ab, nur das hier der Ciffretext verodert wird.
Die Wahl des Initialisierungsvektor muss unvorhersagbar sein, das heißt nicht das die Initialisierungsvektoren selber geheim bleiben müssen (die Sicherheit des Chiffretextes garantiert der Schlüssel) sondern dass die Wahl zukünftiger Initialisierungsvektoren nicht

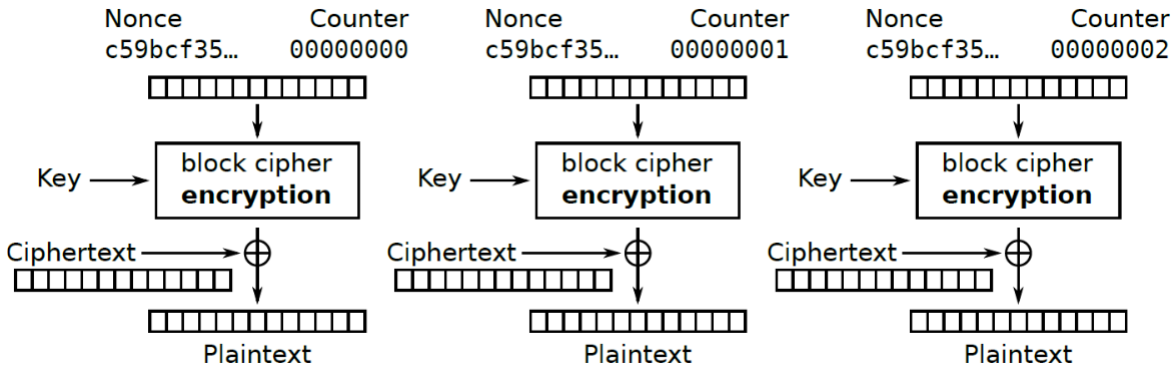


Abbildung 9: Entschlüsselung im CTR Modus, Nonce ist der Initialiserungsvektor

vorhersagbar sein dar (sie also von geheimen Informationen abhängen) und ein Angreifer die Wahl nicht beeinflussen kann[1].
Möglichkeiten zur Wahl des Initialisierungsvektor sind:

- Zufällige Initialisierungvektoren, d.h. eine zufällige Bitfolge der Länge n, hier muss die Bitfolge aber ein Entropie von 95 Bit besitzen ($n \geq 95$?)
- Verschlüsselte Initialisierungsvektoren, wir wählen einen determistisch erzeugten Wert und verschlüsseln ihn mit der einzusetzenden Blockchiffre und Schlüssel, der Chiffretext ist der Initialisierungsvektor

Der randomatisierte Zähler wird als eine Funktion $ctr : \mathbb{Z}_2^* \times \mathbb{N}_0 \to \mathbb{Z}_2^n$ definiert welche die basierend auf dem gegebenen Zufallswert eine Erhöhung in unspezifierter Weise ausführt. Mögliche Implementationen umfassen:

- Einfacher Zähler, welcher den Zufallswert um n erhöht
- Die Hälfte der Bits des Zufallswertes sind reserviert und nur die rechte Hälfte der Bits wird erhöht und läuft über[2]

Das wiederverwenden des Zufallswertet kompromitiert die Sicherheit der Verschlüsselung, da sobald ein Ciffretext einen Klartext zugeordnet wurde der Schlüssel ermittelt werden kann.
Das CTR Modus Kryptosystem wird formal so bezeichnet $((\mathbb{Z}_2^n)^*, \mathbb{Z}_2^n, (\mathbb{Z}_2^n)^*, \mathbb{Z}_2^*, e^*, e^*)$ wobei e und d wie folgt definiert sind:

$$e^* : (\mathbb{Z}_2^n)^* \times \mathbb{Z}_2^m \times \mathbb{Z}_2^n \to (\mathbb{Z}_2^n)^*, e^*(x_0x_1...x_l, k, r) = y_0y_1...y_l \text{ mit } y_i = e(ctr(r,i),k) \oplus x_i$$
$$d^* : (\mathbb{Z}_2^n)^* \times \mathbb{Z}_2^m \times \mathbb{Z}_2^n \to (\mathbb{Z}_2^n)^*, d^*(y_0y_1...y_l, k, r) = x_0x_1...x_l \text{ mit } x_i = e(ctr(r,i),k) \oplus y_i$$

3.4.4 Stromchiffren

Stromchiffren sind pseudozufällige Schlüsselströme welche aus dem Schlüsselwert generiert werden und zur Ver- und Entschlüsselung mittels xor mit den Klar-/Chiffretext kombiniert werden. Deshalb muss der Schlüssel sicher erzeugt sein, da die Sicherheit des Verfahrens nur davon abhängt, denn ein Pseudozufallszahlengenerator ist deterministisch. Formal wird ein Stromchiffre Kryptosystem so bezeichnet: $(\mathbb{Z}_2^*, \mathbb{Z}_2^k, \mathbb{Z}_2^*, e, d)$ mit der Funk-

[1] https://tinyurl.com/y5x3bd5j BSI Seite 76
[2] https://tinyurl.com/y27ga7ew NST Seite 18ff, Anhang B

tion:

$$keystream : \mathbb{Z}_2^* \times \mathbb{Z}_2^k \to \mathbb{Z}_2^* \text{ mit } |keystream(x,z)| = |x|$$
$$e(x,z) = d(x,z) = x \oplus keystream(x,z)$$

Hierbei liegt es an der Implementation des keystreams ob dieser auch vom ersten Parameter abhängig ist oder ob die Zufallswert nur aus dem Schlüssel erzeugt werden.

3.5 Asymmetrische Kryptosysteme

Bei asymmetrischen Kryptosystemen werden zum Ver- und Entschlüsseln zwei verschiedene Schlüssel verwendet. Es gibt also ein Schlüsselpaar aus öffentlichen Schlüssel,der zum Verschlüsseln verwendet wird und einem privaten Schlüssel welcher zum Entschlüsseln verwendet wird. Als asymmetrisches Kryptosystem bezeichen wir das 7-Tupel:

$$(\mathcal{M}, \mathcal{K}_s, \mathcal{K}_p, \mathcal{K}, \mathcal{C}, e, d)$$

Asymmetrische Kryptosysteme sind viel langsamer und benötigen wesentlich größere Schlüssel als symmetrische Kryptographie. Die Kombination beider Verfahren wird als hybride Verschlüsselung bezeichnet, hier wird die Nachricht mit einem symmetrischen System verschlüsselt, die Schlüssel aber über ein asymmetrisches System verschlüsselt, die Nachricht ist hier allerdings nun von der Sicherheit zweier Kroptosystemen abhängig.

3.5.1 RSA-Kryptosystem

Das RSA Verfahren beruht auf der Annahme, dass es kein effizientes Verfahren zum Faktorisieren gibt, das Verfahren kann mit Quantencomputern gebrochen werden. Das Verfahren sollte zur Sicherheit mindestens mit 2048 Bit großen Schlüsseln arbeiten, besser mit 4096 Bit.

1. Wähle große Primzahlen p, q mit $p \neq q$
2. Berechne $n = pq$ und $\varphi(n) = (p-1) \cdot (q-1)$
3. Wähle einen Verschlüsselungsexponenten $1 \neq e \in \mathbb{Z}^\times_{\varphi(n)}$ so dass $\text{ggT}(e, \varphi(n)) = 1$ gilt
4. Berechne den Entschlüsselungsexponenten $d \in \mathbb{Z}^\times_{\varphi(n)}$, so dass
   d das multiplikative Inverse von e ist, d.h.$ed \bmod \varphi(n) = 1$

Hierbei ist (e, n) der öffentliche Schlüssel und (d, n) der private Schlüssel. Zur Verschlüsselung wird anschließend der Klartext in Zahlenwerte umgewandelt, sodass $c = m^e \bmod n$ den Chiffretext und $m = c^d \bmod n$ den Klartext ergibt.
Zur Sicherheit sollten anschließend die Werte $p, q, \varphi(n)$ gelöscht werden, da bereits der Besitz eines einzigen davon das Verfahren bricht, aus p bzw. q lässt sich die andere Primzahl berechnen, da das Produkt beider bekannt ist und mit $\varphi(n)$ lässt sich einfach mit dem öffentlichen Schlüssel der private berechnen.
Eine häufige Wahl für den public-key e ist die Zahl 65537, da dieser von einigen Autoritäten vorgeben wird und ein guter Mittelwert für die Verschlüsselungsgeschwindigkeit darstellt[3].

[3] https://tinyurl.com/y7uy76b9 Crypto Stackexchange

Beispielrechnung:
Wir wählen die Primzahlen $p = 47$ und $q = 59$ und als Nachricht den Wert $m = 2345$.

$$n = p \cdot q = 47 \cdot 59 = 2773$$
$$\varphi(n) = (p-1)(q-1) = 46 \cdot 58 = 2668$$

Damit haben wir also $\mathbb{Z}_{2668}^{\times}$ und können z.B. $e = 17$ wählen. Damit ergibt sich für d mithilfe des erweiterten euklidischen Algorithmus:

$\varphi(n)$	e	$\lfloor \frac{\varphi(n)}{e} \rfloor$	x	y
2668	17	156	-1	1-156·(-1)=157
17	16	1	1	0-1·1=−1
16	1	16	0	1
1	0		1	0

Damit ist der $d = 157$ und der private Schlüssel also $(157, 2773)$ und der öffentliche Schlüssel $(17, 2773)$ Damit ergibt sich:

$$c = m^e mod n = 2345^{17} mod 2773 = 2030$$
$$m = c^d mod n = 2030^{157} mod 2773 = 2345$$

3.5.2 Elgamal-Kryptosystem

Ein Verfahren was auf der Annahme beruht, dass der diskrete Logarithmus nicht einfach berechenbar ist.
Die Schlüsselerzeugung funktioniert wie folgt:
1. Wähle eine zyklische[4] Gruppe $\mathcal{G} = (G, \diamond, e)$ mit Erzeuger g und $a \in \{2, ..., ord(\mathcal{G})[5] - 1\}$, setze $A = g^a$
2. Privater Schlüssel ist $(\mathcal{G}, g, a)$
3. Öffentlicher Schlüssel ist $(\mathcal{G}, g, A)$

Die Verschlüsselung erfolgt wie folgt:
1. Wähle zufällig $r \in \{1, ..., ord(\mathcal{G}) - 1\}$, setze $R = g^r$
2. Berechne $K = A^r = (g^a)^r = g^{ar}$ und $C = m \diamond K$
3. Sende $e(m, (\mathcal{G}, g, A)) = (R, C)$

Die Entschlüsselung funktioniert wie folgt:
1. Berechne $K = R^a = (g^r)^a = g^{ra} = g^{ar}$
2. Bestimme K^{-1} in $\mathcal{G}$
3. $d((R, C), (\mathcal{G}, g, a)) = C \diamond K^{-1} = m$

3.6 Hashfunktionen

Mittels Hashfunktionen können die Schutzziele Integrität und Authentizität, sowie Nicht-Abstreitbarkeit gewährleistet werden.
Sei A ein Alphabet und $m, n \in \mathbb{N}$ mit $n < m$, dann heißt die Funktion $h : A^m \to A^n$ Kompressionsfunktion und $h : A^* \to A^n$ Hashfunktion. Diese Funktionen heißen schwach

[4] Sie besitzt einen Erzeuger, sodass $\langle g \rangle := \{g^n : n \in \mathbb{Z}\} = \mathcal{G}$ (Potenz mit dem gegebenen Verknüpfungsoperator)
[5] Anzahl der Elemente in Gruppe bzw. unendlich

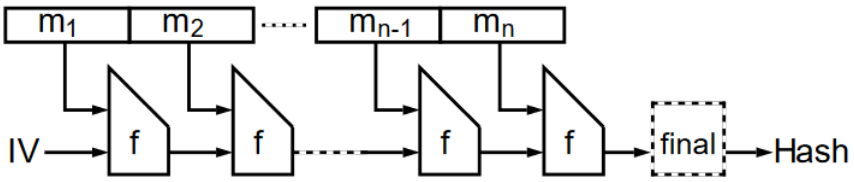


Abbildung 10: Bei Merkle-Damgård wird jeder Block mit den vorigen Block komprimiert und dann mit g finalisiert

kollisionsresistent, falls kein Angreifer in der Lage ist effizient zu einem gegebenen Urbild x einen zweiten Wert x' zu finden der auf den selben Wert abbildet,d.h. $x, x' \in A : x \neq x' \wedge h(x) = h(x')$. Sie heißen stark kollisionsresitent, falls es nicht möglich ist irgendeine Kollision effizient zu bestimmen.
Anforderungen an kryptographische Hashfunktionen sind:

- Leicht und schnell zu berechnen
- Einwegfunktion
- (Stark) kollisionsresistent
- Kleine Änderungen an der Eingabe haben große Änderungen am Hashwert

Eine Kompressionsfunktion kann auch als Konkatenation der Verschlüsselungsfunktion von Blockchiffren aufgefasst werden.

3.6.1 Merkle-Damgård-Konstruktion

Sei A ein Alphabet, $f : A^{n+m} \to A^n$ eine Kompressionsfunktion,
pad:$A^* \to (A^m)^*$ eine Auffüllfunktion, $x_0 \in A^n$ ein beliebiger Initialisierungsvektor und
g:$A^n \to A^n$ eine Finalisierungsfunktion.
Dann ist die Hashfunktion $h : A^* \to A^n$ für $x \in A^*$ definiert durch

1. $x_1 x_2 ... x_k = pad(x)$ mit $x_i \in A^m$
2. $h_0 = f\,(conc(x_0, x_1))$ und $h_i = f\,(conc(h_{i-1}, x_i))$ für $1 \leq i \leq k$
3. $h(x) = g(h_k)$

Wobei das Salt oft $x_0 = 0^n$ als 0-Padding und $g = id_{A^n}$ als Identitätsfunktion gewählt wird.

3.6.2 Exkurs: Chosen Target Forced Prefix

Die Merkle-Damård-Konstruktion hat gegebenfalls - falls die Kompressionsfunktion f nicht schwach kollisionsresistent ist - Schwächen für Chosen Target Forced Prefix (CTFP) Angriffe, wo ein Angreifer für einen gegebenen Hash H einer nicht umbedingt geheimen Nachricht N, aus einem erzwungenen Präfix P kokateniert mit einem frei wählbaren Suffix S eine Nachricht erzeugt, sodass $h(M) = H = h(PS)$ gilt. Hat der Angreifer eine freie Wahl über die möglichen Formulierungen von P, sodass deren Bedeutung erhalten bleibt, kann hier nun aus einer signifikant kleineren Menge an Kodierungen ein Suffix S gesucht werden, was den Angriff leichter macht.
Darauf aufbauend lässt sich ein Herding-Angriff[6] konstruieren:

[6] https://eprint.iacr.org/2005/281.pdf Herding Hash Functions

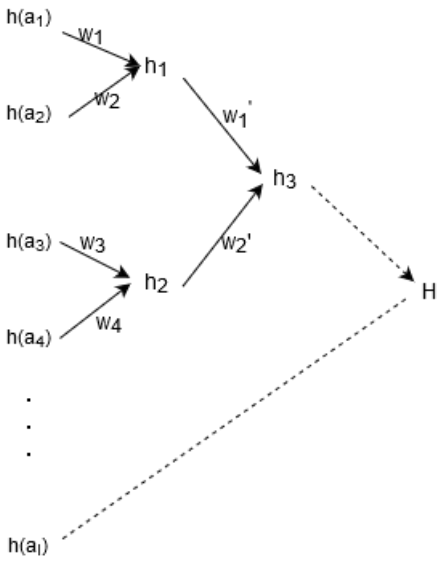


Abbildung 11: Dieser Hashbaum gibt Pfade an um aus 2^k Hashes den gewünschten Hash durch Konkatenation der Strings zu erreichen.

1. Konstruieren einer Baumstruktur, sodass für 2^k Startblöcke unseres Suffixes die Kanten Strings enthalten, sodass jeweils 2 Knoten zu einem Zwischenhash h_i zusammenlaufen. Dies wird solange wiederholt bis alle Pfade zu einem finalen Hash H zusammenlaufen welcher commited wird. Um eine Stufe abzusteigen werden rund
2. Nach einiger Zeit, wird der zu wählende Präfix P bekannt/gewählt
3. Wir suchen nun einen einzigen Block, sodass wenn dieser an P konkateniert wird einen Zwischenhash in unser Baumstruktur ergibt.
4. Nun werden die Strings bis hin zur Wurzel konkateniert, wodurch eine gefälschte Nachricht entsteht die den selben Hash besitzt wie die unsprünglich committete

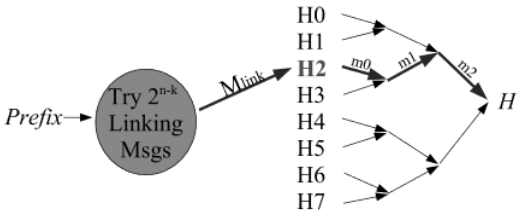


Abbildung 12: Nachdem ein Linkender String gefunden wurde, kann die gefälschte Nachricht zusammengesetzt werden

Ist die Menge an möglichen Präfixen im vorhinein bekannt, so kann die Menge an Starthashes entsprechend dieser Präfixe gewählt werden, wodurch der Aufwand für die Suche des Linkenden Strings gespart wird. Ist die Menge so groß dass nicht alle möglichen Hashkombinationen berechnet werden können, so könnte auch das Risiko einer Entarnung in Kauf genommen werden, wenn ein genügend ähnlicher Präfix verwendet wird. Des weiteren können auch sofern die zu verändernden Stellen innerhalb von abzugrenzenden Blöcken eines Dokumentes liegen, für jede einzelne Stelle eigene alternierende Texte mit den selben Hashes gefunden werden, sodass diese auch ohne eine Baumstruktur vertauscht werden können.

3.6.3 HMAC-Konstruktion

Sei $H : \mathbb{Z}_2^* \to \mathbb{Z}_2^{8n}$ eine Hashfunktion, $opad = (0x5C)^n = (01011100)^n \in \mathbb{Z}_2^{8n}$
und $ipad = (0x36)^n = (00110110)^n \in \mathbb{Z}_2^{8n}$, dann definieren wir $HMAC : \mathbb{Z}_2^* \times \mathbb{Z}_2^* \to \mathbb{Z}_2^*$ als

$$HMAC(x, k) = H\left(conc\left(K \oplus opad, H\left(conc\left(K \oplus ipad, x\right)\right)\right)\right)$$

Dabei ist die Wahl von $K \in \mathbb{Z}_2^{8n}$ von k abhängig:

- $|k| = 8n \Rightarrow K = k$
- $|k| = m < 8n \Rightarrow K = k0^{8n-m}$, d.h. k wird auf Blockgröße aufgefüllt
- $|k| > 8n \Rightarrow K = H(k)$

Der Keyed-Hash Message Authentification Code (HMAC) hashed hierbei die Nachricht x, sodass auch bei der Verwendung einer Hashfunktion welche nicht kollisionsresistent ist das Verfahren nicht unsicher wird. Da der Hashwert der Nachricht direkt innerhalb einer Hashoperation berechnet wird, gilt die Grundlage für CTFP nicht, da hier die Berechnung nicht in unterteilten Blöcken funktioniert. $H(x) = H(x') \nRightarrow H(xS) = H(x'S)$
Damit ist die HMAC Konstruktion zwar effizient und eine Fälschung einer Nachricht schwierig, um die Authentizität einer Nachricht zu prüfen müssen aber beide Parteien den geheimen Schlüssel kennen um den Hash neu zu bilden und zu prüfen.
Varianten des HMAC sind:
HMAC-based One-time Password Algorithmus (HOTP) verwendet einen Zähler welcher nach jeder Hashgeneration inkrementiert wird und bei der Hashgeneration den Schlüsssel erhöht. Anschließend wird der Hash in einen 6-10 stelligen numerischen Wert umgewandelt welcher auch von Menschen lesbar ist. Die selben Informationen müssen dem Authentifizierenden vorliegen, sodass falls dieser den Hash reproduzieren kann die Nachricht als gültig anerkannt wird.
Time-based One-time Password Algorithmus (TOTP) verwendet anstelle eines Zählers einen Zeitstempel, funktioniert sonst aber wie HOTP. Um die Ungenauigkeiten nicht synchronisierter Uhren auszugleichen umfasst der Gültigkeitsraum eines TOTP Hashes oft rund 30 Sekunden, bevor der Hash ungültig wird. Das Verfahren kommt vor allem in der 2-Faktor Authentifizierung zum Einsatz.

3.7 Digitale Signaturen

Signaturen erlauben es die Authentizität, Integrität und Nicht-Abstreitbarkeit einer Nachricht zu testen bzw. sicherstellen.
Ein Angreifer kann keinen Schlüssel generieren, sodass damit für zwei verschiedene Nachrichten die selbe Signatur erzeugt wird.
Wir unterscheiden:

- Fortgeschrittene elektronische Signatur
  - Signatur ist eindeutig dem Unterzeichner zuordenbar
  - Ermöglicht Identifizierung des Unterzeichners
  - Wird unter Verwendung elektronischer Signaturerstellungsdaten erstellt, die der Unterzeichner mit einem hohen maß an Vertrauen unter seiner alleinigen Kontrolle verwenden kann
  - Nachträgliche Veränderung unterzeichneter Daten muss erkannt werden

- Qualifizierte elektronische Signatur
  - Rechtliche Gleichstellung mit handschriftlicher Unterschrift
  - Eine Signatur welche auf einem von einem Mitgliedsstaat ausgestellten Zertifikat beruht, ist in allen Mitgliedsstaaten gültig

3.7.1 Wissen und Ziele von Angreifern

Wie viel Wissen ein Angreifer hat unterscheiden wir mit folgenden Begriffen:

- Key-Only Attack, hier besitzt der Angreifer nur den öffentlichen Schlüssel
- Known Signature Attack, hier sind mehrere Paare von Nachricht und Signatur bekannt
- Chosen Message Attack, der Angreifer kann im Vorfeld für beliebige selbstgewählte Nachrichten die Signatur bekommen
- Adaptive Chosen Message Attack, kann während des Angriffes für beliebig viele selbstgewählte Nachrichten zugehörige Signaturen bekommen

Die Ziele eines Angreifers mit folgenden Begriffen:

- Existential Forgery, der Angreifer will ein gültiges Nachricht/Signatur-Paar ermitteln
- Selective Forgery, gültige Signatur zu einzelnen neuen, vor dem Angriff bekannten Nachrichten ermitteln
- Universal Forgery, gültige Signatur zu jedem beliebigen Dokument ermitteln
- Total Break, Angreifer bestimmt den geheimen Schlüssel

Der stärkste Sicherheitsbegriff ist der des *Existential forgery under a chosen message attack*

Ein Signaturschema gilt als **sicher**, wenn jeder Angreifer nach dem er n Nachrichten signiert hat mit **sehr hoher Wahrscheinlichkeit** keine Nachricht findet, sodass diese von den ursprünglich n verschieden ist aber die Signatur mit einer davon übereinstimmt

3.7.2 Signieren mit RSA

Mittels RSA kann signiert werden, denn die Ver-/Entschlüsselung kommutieren. Dazu wird zusätzlich noch eine Hashfunktion $h : A* \to \{0, ..., n\}$ benötigt.
Die Signatur wird wie folgt erstellt: $s = D(h(m), (d, n)) = (h(m))^d \mod n$
Überprüfen lässt sie sich wie folgt: $h(m) = E(s, (e, n)) = s^e \mod n$

Beispiel Signaturvorgang mit RSA:
Signieren:

1. Schlüsselgenerierung mit $(d, n) = (37313, 241103)$ und $(e, n) = (65537, 241103)$
2. Alice will die Nachricht m mit $h(m) = 164796$ signieren
3. Sie berechnet $s = h(m)^d \mod n = 164796^{37313} \mod 241103 = 129335$

Verifizieren:

1. Bob besitzt (e, n), sowie die Nachricht m und ihre Signatur $s = 129335$

2. Er berechnet $s^e \bmod n = 129335^{65537} \bmod 241103 = 164796$
3. Er berechnet den Hash der Nachricht m
4. Da beide Berechnungen das selbe Ergebnis haben, akzeptiert Bob die Signatur

Eine Signaturerstellung mit RSA ist nur sicher, sofern RSA sicher ist.

3.7.3 Digital Signature Algorithm

Parametergenerierung:

1. Wähle gleichverteilt zufällig eine große Primzahl q (160 Bit, 224 Bit, 256 Bit)
2. Wähle große Primzahl p (1024 Bit, 2048 Bit, 3072 Bit), sodass $q|(p-1)$ gilt
3. Suche ein $g \in \mathbb{Z}_p^\times$ mit ord[7]$(g) = q$,d.h. wir wählen $h \in \mathbb{Z}_p^\times$ mit $g := h^{\frac{p-1}{q}} \bmod p \neq 1$.[8]
4. Die Parameter p, q, g sind öffentlich

Schlüsselgenerierung:

1. Wähle zufällig x mit $1 < x < q$
2. Berechne $y = g^x \bmod p$

Dann ist y der öffentliche Schlüssel und x der private Schlüssel.
Die Berechnung des privaten Schlüssel aus dem öffentlichen Schlüssel ist eine Instanz des diskreten Logarithmus, und somit nicht effizient lösbar.

Signieren:

1. Wähle k mit $1 < k < q$
2. Bestimme $r = (g^k \bmod p) \bmod q$. Falls $r = 0$, beginne von vorne
3. Bestimme $s = k^{-1} \cdot (H(m) + r \cdot x) \bmod q$. Falls $s = 0$, beginne von vorne
4. Signatur ist das Tupel (r, s)

Verifikation:

1. Ungültig, wenn $0 < r < q$ oder $0 < s < q$ nicht gilt
2. Berechne $w = s^{-1} \bmod q$
3. Berechne $u_1 = H(m) \cdot w \bmod q$
4. Berechne $u_2 = r \cdot w \bmod q$
5. Berechne $v = (g^{u_1} \cdot y^{u_2} \bmod p) \bmod q$
6. Akzeptiere die Signatur, falls $v = r$

[7]Die Ordnung eines Elementes ist die Zahl $\text{ord}(g) := \inf\{n \in \mathbb{N} : g^n = e\} \in \mathbb{N} \cup \{\infty\}$

[8]Da $ggT(h, p) = 1$ ist, gilt $g \equiv_p h^{\frac{p-1}{q}} \xRightarrow{\text{Kleiner Fermat}} g^q \equiv_p h^{p-1} \equiv_p 1 \Rightarrow \text{ord}(g) \leq q$.
Da q prim ist kann die Ordnung von g kein Teiler von q sein, damit gilt $\text{ord}(g) = q$

Beispiel für DSA
Parametergenerierung:

1. Wir wählen $q = 59$ und $p = 709$
2. Mit $h = 5$ gilt $g := 5^{\frac{709-1}{59}} \bmod 709 = 20 \neq 1$
3. Also sind $(p, q, g) = (709, 59, 20)$ die öffentlichen Parameter

Schlüsselgenerierung:

1. Wähle $x = 23$ als privaten Schlüssel
2. Bestimme $y = g^x \bmod p = 20^{23} \bmod 709 = 186$ als öffentlicher Schlüssel

Signieren:
Wir besitzen $p = 709, q = 59, g = 20$ und $x = 23$, der Hashwert unserer Nachricht ist $H(m) = 16$

1. Wir wählen $k = 36$
2. Wir bestimmen $r = (20^{36} \bmod 709) \bmod 59 = 31$
3. Mit $k^{-1} = 41$ bestimmen wir $s = k^{-1} \cdot (H(m) + r \cdot x) \bmod q = 29889 \bmod 59 = 35$
4. Die Signatur ist damit $(r, s) = (31, 35)$

Verifizieren:
Es sind $p = 709, q = 59, g = 20, y = 186, H(m) = 16$ und der Signatur $(r, s) = (31, 35)$

1. r und s sind gültig
2. Wir berechnen $w = s^{-1} \bmod q = 27$
3. Wir berechnen $u_1 = H(m) \cdot w \bmod q = 16 \cdot 27 \bmod 59 = 19$
4. Wir berechnen $u_2 = r \cdot w \bmod q = 31 \cdot 27 \bmod 59 = 19$
5. Wir berechnen $v = (g^{u_1} \cdot y^{u_2} \bmod p) \bmod q = (20^{19} \cdot 186^{11} \bmod p) \bmod q = 31$
6. Da $v = r$ gilt, wird die Signatur akzeptiert

3.8 Schlüsselverteilung

Wir unterscheiden verschiedene Schlüssel:
Langzeitschlüssel haben eine lange Gültigkeit (1 Monat bis 1 Jahr) und werden häufig zur Authentifizierung verwendet.
Kurzzeitschlüssel bzw. Sitzungsschlüssel haben nur eine begrenzte Gültigkeit und minimieren somit das Risiko falls der Schlüssel kompromittiert wird.

Ob bei einem asymmetrischen System ein öffentlicher Schlüssel tatsächlich von einer bestimmten Person kommt, kann mithilfe von Zertifkaten welche von einem vertrauenswürdigen Dritten - Public Key Infrastructure (PKI) - ausgestellt werden, verifiziert werden. Ein solches Zertifikat beinhaltet unter Anderem: Öffentlicher Schlüssel, Name, Gültigkeitszeitraum, Austeller,...
Das Zertifikat wird dann mit einer digitalen Signatur des Ausstellers versehen, sodass derjenige der die Authentizität des Schlüssels prüfen will trotzdem noch dem Schlüssel des Zertifikat-Ausstellers vertrauen muss oder eben der nächst höheren Instanz welche den Schlüssel des Zertifikats-Austellers signiert.

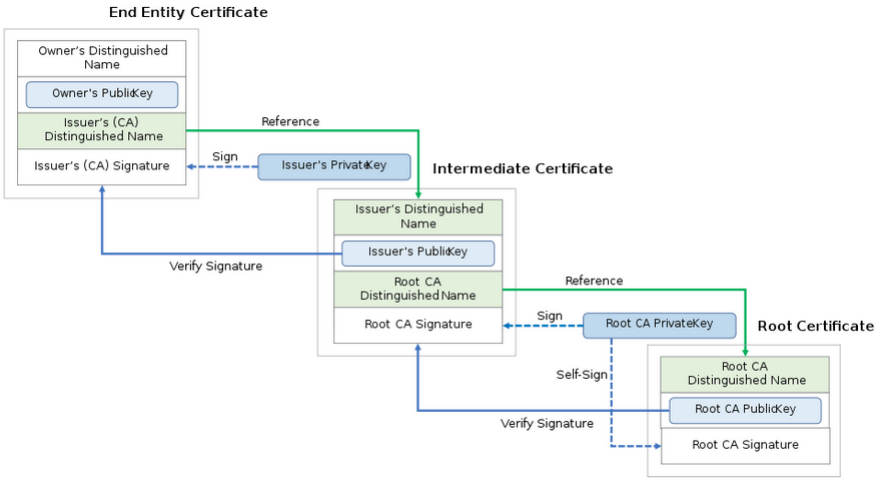


Abbildung 13: Ein zentralisierter Ansatz ist X.509 als PKI, Root Certificates muss vertraut werden

Ein dezentralisierter Ansatz ist PGP, hierbei werden Schlüsselbünde versandt, sodass der Empfänger zum einen weiß welchen anderen öffentlichen Schlüsseln der Sender vertraut und dann in seinen bereits vorher empfangenen Schlüsselbändern nachsehen kann ob andere Personen denen er bereits vertraut auch dem Sender vertrauen. Falls dies der Fall ist kann er die neu erhaltenen Schlüssel als vertrauenswürdig speichern, sodass über Zeit ein Web of Trust entsteht wo alle Parteien untereinander ihre öffentlichen Schlüssel kennen. Hierbei ist aber fraglich ob Vertrauen wirklich transitiv ist.

Ein Problem bei beiden Ansätzen sind das ein Schlüsselverlust/-kompromittierung dafür sorgt dass die Nachrichten nicht mehr lesbar/geheim sind.
Als Lösung für letzteres kann ein Ablaufdatum für Zertifikate festgelegt werden oder mittels Widerrufszertifikaten vorher ausgestellte Zertifikate widerrufen werden. Diese Widerrufszertifikate können bei X.509 von einem zentralisierten Server ausgestellt werden, wobei auch hier die Vertrauenswürdigkeit und Erreichbarkeit des Servers fragwürdig ist. Beim Web of Trust können diese durch die Benutzer oder angegebene Parteien erstellt und verbreitet werden. Diese Updates zu verbreiten ist aber ineffizient, sollen alle alles speichern?

3.9 Schlüsselaustausch

Beim Man-in-the-middle Angriffsmodel hat der Angreifer vollen Zugriff auf dem Kommunikationskanal. Er kann daher:

- Nachrichten abfangen
- Übermittlung von Nachrichten verzögern
- Nachrichten unterdrücken
- Nachrichten durch andere Nachrichten ersetzen
- Nachrichten unter falscher Identität senden
- Er kann aber keine kryptographische Primitive nicht brechen

Das naive Schlüsselaustausch Protokoll kann um eine PKI erweitert werden, welche die öffentlichen Schlüssel von A und B kennt. A holt von T den öffentlichen Schlüssel K_B

Abbildung 14: Man-in-the-middle Angriff, wo A einen neuen symmetrischen Schlüssel, mit öffentlichem Schlüssel K_B von B verschlüsselt an B sendet, die Chiffre wird vom Angreifer abgefangen

von B und verschlüsselt damit den neuen Sitzungsschlüssel und schickt diesem zusammen mit seinem Namen an B. Auch hier kann ein Angreifer die Chiffre abfangen und seinen Schlüssel in Verbindung mit seinem Namen an B senden.

3.9.1 Needham-Schroeder Protokolle

Die **symmetrische Variante** des Protokolls läuft wie folgt ab:

(1) $A \rightarrow T : A, B, N_A$
(2) $T \rightarrow A : \{N_A, K, B, \{K, A\}_{K_B}\}_{K_A}$
(3) $A \rightarrow B : \{K, A\}_{K_B}$
(4) $B \rightarrow A : \{N_B\}_K$
(5) $A \rightarrow B : \{N_B - 1\}_K$

Hierbei sind N_A und N_B zufällig generierte Nonces. Die in Klammern geschriebene Werte werden als eine zusammengesetzte Nachricht mit dem tiefgestellten Schlüssel verschlüsselt. Dabei muss T im Besitz eines geheimen Schlüssels K_A und K_B zum kommunizieren mit jeweils A und B sein.

Indem A im ersten Schritt die gewünschten Kommunikationspartner und eine Nonce N_A an den Server sendet kann dieser einen geheimen Sitzungsschlüssel K generieren über den nach Abschluss des Verfahrens eine gesicherte Kommunikation zwischen A und B möglich wird. Damit die im zweiten Schritt von A empfangene Chiffre auch tatsächlich einen neuen Sitzungsschlüssel enthält muss N_A neu sein, da sonst ein Angreifer eine aufgezeichnete Nachricht anstelle von T an A senden könnte um die Verwendung eines alten Schlüssels zu erzwingen. Damit außerdem kein im ersten Schritt der Empfänger nicht durch einen anderen ausgetauscht werden kann ohne das A es bemerkt sendet T auch B erneut zurück. Die innere mit K_B verschlüsselte Chiffre enthält das in Schritt 3 an B gesendete Paket und kann nur von B entschlüsselt werden. Damit sich auch B davon überzeugen kann dass der Sitzungschlüssel K neu ist sendet er eine mit K verschlüsselte Nonce N_B, welche A dekrementiert und zurücksendet, da N_B neu ist kann auch B sicher sein, dass es sich um keinen Replay-Angriff handelt[9].
Leider obliegt die Schlüsselgenerierung dem Knoten T auch kennt T nun den Sitzungsschlüssel.

[9]Eine Angriffsform wo ein Angreifer zuvor aufgezeichnete Daten sendet um die Identität des vorherigen Gesprächpartners vorzutäuschen

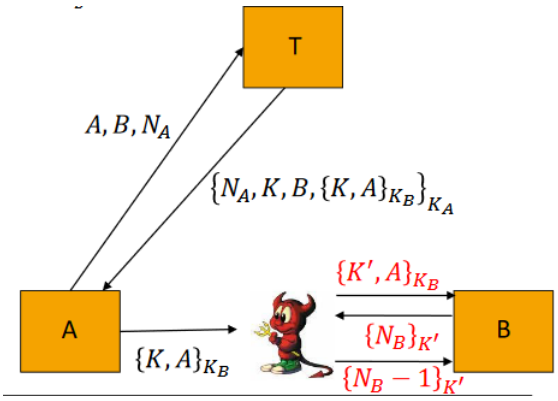


Abbildung 15: Die symmetrische Variante des Needham-Schroeder-Schlüsselaustausches, falls der Angreifer ein altes Paar $\{K', A\}_{K_B}$ und K' kennt kann der Angreifer die Indentität von A annehmen

In der **antisymmetrische Variante** funktioniert der Datenaustausch wie folgt:

(1) $A \to T : A, B$
(2) $T \to A : \{K_{PB}, B\}_{K_{ST}}$
(3) $A \to B : \{N_A, A\}_{K_{PB}}$
(4) $B \to T : B, A$
(5) $T \to B : \{K_{PA}, A\}_{K_{ST}}$
(6) $B \to A : \{N_A, N_B\}_{K_{PA}}$
(7) $A \to B : \{N_B\}_{K_{PB}}$

Hierbei sind K_{PA} und K_{PB} die öffentlichen Schlüssel von A bzw. B, sowie K_{ST} der private Signaturschlüssel von T.

3.9.2 Diffie-Hellman-Protokoll

Zu Beginn muss eine Primzahl p und ein Generator $g \in \mathbb{Z}_p^\times$ für eine Gruppe $\mathbb{Z}_p$ gemeinsam gewählt werden. Dann läuft das Verfahren wie folgt ab:

1. A wählt $a \in \mathbb{N}$ mit $0 < a < p$ zufällig und sendet $g^a \bmod p$ an B
2. B wählt $b \in \mathbb{N}$ mit $< b < p$ zufällig und sendet $g^b \bmod p$ an A
3. A berechnet $(g^b)^a \bmod p$
4. B berechnet $(g^a)^b \bmod p$

Damit haben A und B den selben Schlüssel g^{ab} und können darüber kommunizieren. Ein passiver Angreifer, d.h. ein Angreifer welcher nur den Datenaustausch mithören kann a oder b nicht einfach berechnen.
Allerdings wurde früher oft nur eine geringe Menge an Primzahlen für p verwendet, sodass die möglichen Logarithmen vorberechnet werden konnten. Zwei Vorberechnungen deckten 66% der IPsec-VPNs und 26% der SSH-Server ab. Bei einem MitM-Angriff kann der Angreifer damit parallel zwei DH-Protokolle laufen lassen und so Schlüssel selber wählen. Als Lösung hierfür können die Nachrichten signiert werden authenticated DH/Station-to-Station-Protokoll.

Für das Station-to-Station-Protokoll (STS) müssen beide Parteien einen Signaturschlüssel sk_A und sk_B besitzen und die Zertifikate der Schlüssel sind beiden bekannt. Wir bezeichnen $K = g^{ab}$

(1) $A \to B : g^a$
(2) $B \to A : g^b, \{sig(sk_B, (g^a, g^b))\}_K$
(3) $A \to B : \{sig(sk_A, (g^a, g^b))\}_K$

3.10 Ausblick

3.10.1 Integritätsschutz mit Verschlüsselung

Man unterscheidet zwischen Authenticated Encryption (AE), wo Daten verschlüsselt werden und diese Chiffre nicht manipulierbar sein soll, und Authenticated Encryption with Associated Data (AEAD), wo zusätzlich zu AE noch Metadaten unverschlüsselt übertragen werden sollen deren Integrität auch sichergestellt werden muss.

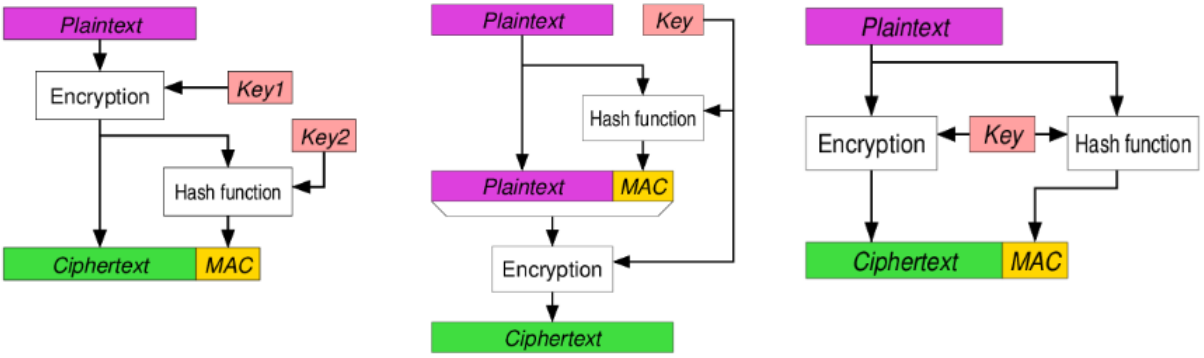


Abbildung 16: Dies sind AE-Verfahren: EtM ist sicher, MtE und E&M sind nur bedingt sicher

Ein AEAD Verfahren ist Galois/Counter-Mode welches mit 128 Bit Blockchiffren arbeitet und Integritätsschutz für verschlüsselte Daten bietet.

- $GCTR_K(ICB, X_1||X_2||...||X_n^*) = Y_1||Y_2||...||Y_n^*$
  1. Falls X leer ist, gibt es zurück
  2. Setzte den ersten Zählblock $CB_1 = ICB$ auf den Initialisierungswert
  3. Für i = 1 bis n sei $CB_i = inc_{32}(CB_{i-1})$
  4. Für i = 1 bis n-1 sei $Y_i = X_i \oplus CIPH_K(CB_i)$ für i = n wir bis zum Blockende xored, sodass Y_n^* selbe Länge wie X_n^* hat.
- $GHASH_H(X_1||X_2||...||X_m) = Y_m$
  1. Sei $Y_0 = 0^{128}$
  2. Für $i = 1, ..., m$ sei $Y_i = (Y_{i-1} \oplus X_i) \bullet H$, wobei $\bullet$ der Multiplikation im Galois Körper $\mathbb{F}_{2^{128}}$ entspricht.

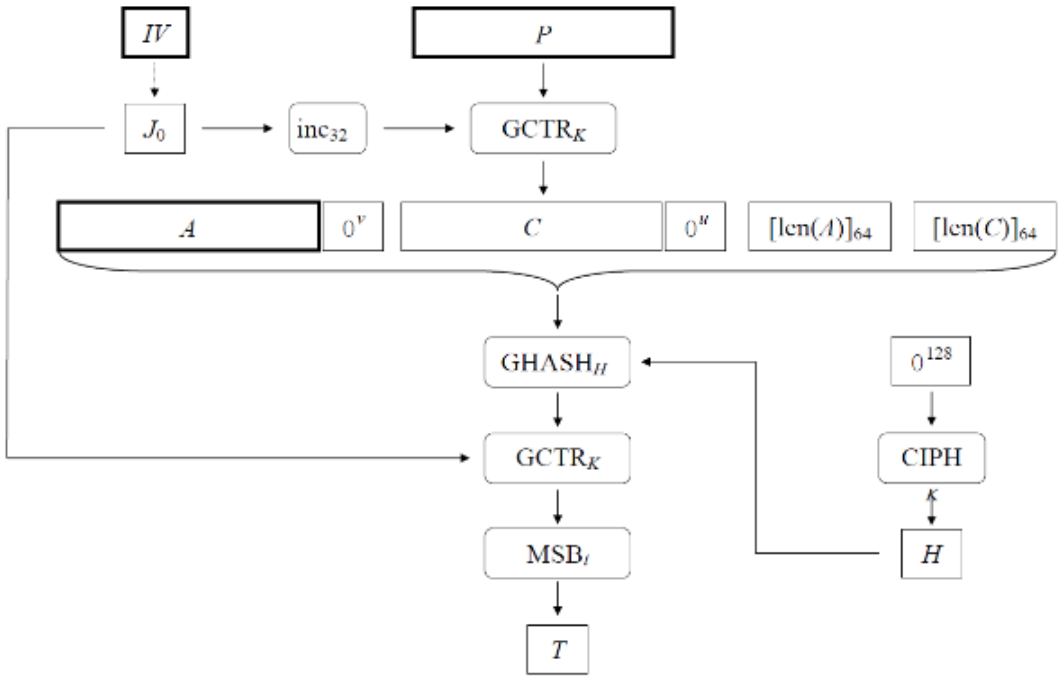


Abbildung 17: Im GCM-Verfahren wird

3.10.2 Secret Sharing

Man möchte Informationen so verschlüsseln, dass man zum Entschlüsseln eine gewisse Gruppe von Leuten zusammenarbeiten müssen. Eine Variante ist das (t, n)-Schwellenwert-Kryptosystem wo mindestens t von insgesamt n Shareholdern zusammenarbeiten müssen um eine Nachricht zu entschlüsseln.

Ist Geheimhaltung wichtiger als Verfügbarkeit sollte t nah an n gewählt werden, anders herum sollte t möglichst klein aber größer als $\frac{n}{2}$ gewählt werden.
Um ein (t, n)-Kryptosystem zu konstruieren wendet man $\binom{n}{t}$-mal das (t, t)-Schwellenwert-Kryptosystem an. Damit bekommt jeder Shareholder $\binom{n-1}{t-1}$ Shares, um jeden möglichen Ausfall von $(n - t)$ Shareholdern abzudecken.
Hierfür nehmen wir an, dass der Dealer vertrauenswürdig und der Kommunikationskanal zwischen Dealer und Shareholder, und Shareholder und Combiner sicher sind. Außerdem nimmt man an dass zumindest die Hälfte aller Shareholder ehrlich ist.

Shamirs Secret Sharing Protokoll: Wir haben einen Schlüssel k und wollen das mindestens t Shareholder zusammenarbeiten müssen um ihn wiederherzustellen. Sei n die Anzahl an Shareholdern.
Wahl der Shares:

1. Dealer wählt Primzahl p mit $p > k$ und $p > n$
2. Dealer wählt $t - 1$ Koeffizienten $a_1, a_2, ..., a_{t-1} \in \mathbb{Z}_p$
3. Dealer wählt Polynom $f(x) = k + a_1 \cdot x + a_2 \cdot x^2 + ... + a_{t-1} \cdot x^{t-1}$ mit $f(0) = k$ und $a_{t-1} \neq 0$
4. Die Shares sind $\{s_1, s_2, ..., s_n\} = \{(1, f(1)), (2, f(2)), ..., (n, f(n))\}$

Entschlüsselung:

1. t der n Shareholder geben ihre Shares dem Combiner
2. Der Combiner versucht das Polynom wieder herzustellen (Lagrange-Interpolation)
   (a) Definiere $l_i = \prod_{j=1, j\neq i}^{t} \frac{x-x_j}{x_i-x_j}$
   (b) Bemerke $l_i(x_k) = \begin{cases} 1, & \text{wenn } i = k \\ 0, & \text{wenn } i \neq k \end{cases}$
   (c) Definiere $L(x) = \sum_{i=1}^{t} y_i \cdot l_i(x)$
   (d) Bemerke $L(x_i) = y_i$ für alle $1 \leq i \leq t$

Das Verfahren ist perfekt sicher, selbst wenn man $t-1$ Shares besitzt, sind alle Schlüssel gleich wahrscheinlich. Sofern t gleich bleibt, können weiter Shares ohne Probleme hinzugefügt werden. Bei einem neuen t kann das Verfahren mit dem selben k einfach erneut ausgeführt werden.

3.10.3 Post-Quanten-Kryptographie

Grovers Algorithmus ist ein Algorithmus für Quantencomputer welcher unsortierte Datenbanken in $\mathcal{O}(\sqrt{n})$ Schritten durchsucht. Also findet er Schlüssel der Länge n in $\mathcal{O}(\sqrt{2^n})$ Schritten. Eine einfache Lösung für symmetrische Kryptographie ist eine Vergrößerung (mindest Verdopplung) der Schlüssellänge.

Shors Algorithmus hat zum Ziel n effizient zu faktorisieren.

1. Wähle $a \leq n$ zufällig
2. Wenn $\text{ggT}(a, n) \neq 1$, so gebe $\text{ggT}(a, n)$ aus
3. Finde die Periode r von a modulo n, d.h. die kleinste natürliche Zahl r, so dass $a^r \equiv 1 \bmod n$
4. Wenn r ungerade ist, oder $a^{\frac{r}{2}} + 1 \equiv 0 \bmod n$, so starte erneut bei Schritt 1
5. Nun gilt $a^r - 1 = kn$
6. Gib also $\text{ggT}(a^{\frac{r}{2}} \pm 1, n)$ als Lösung zurück

Der **Algorithmus von Deutsch** kann entscheiden ob eine Funktion $f : \{0,1\} \to \{0,1\}$ konstant ist oder nicht, ohne die unterspezifizierte Funktion zwei mal auswerten zu müssen.

In der **hashbasierte Kryptographie** wird der Hash der Schlüssels als öffentlicher Schlüssel verwendet und sein Urbild als privater Schlüssel, denn die Urbildberechnung ist auch für Quantencomputer schwer.
Lamport-(Diffie-)Einmal-Signaturverfahren (Schlüsselerzeugung): Alice will eine Nachricht der Länge 256 Bit signieren.

1. Alice wählt (zufällig und gleichverteilt) 256 Paare von Zahlen, jeweils 256 Bits lang (privater Schlüssel)
2. Sie hasht nun jede Zahl, die Hashs sind der öffentliche Schlüssel

Signierung:

1. Für jedes Bit der Nachricht wählt Alice je nach Wert des Bits entweder die erste oder die zweite Zahl aus einem Paar ihres privaten Schlüssels
2. Die gewonnene Zahlenfolge ist ihre Signatur

Verifizierung:
Bob hasht jede Zahl der Signatur und prüft, ob die Ergebnisse zu den jeweils passenden Zahlen des öffentlichen Schlüssels passen.

Der Schlüssel darf nur einmal verwendet werden, da ein Angreifer von jeder Signatur lernt und aus Signaturen bekannter Nachrichten auch Signaturen für neue Nachrichten erzeugen kann. Zum Beispiel kann aus den Signaturen zu Nachricht 0010 und 0111 auch die Signatur zu 0110 und 0011 erzeugt werden.
Der öffentliche Schlüssel und die Signatur sind im Verhältnis zur Nachrichtenlänge sehr lang.

3.10.4 Zero-Knowledge Beweise

Bei Zero-Knowledge Beweisen möchte ein Beweiser P einen Verifizierer V davon überzeugen, dass er ein Geheimnis kennt, ohne dabei irgendwelche Informationen über das Geheimnis preiszugeben. Außerdem soll ein außenstehender Dritter nicht davon überzeugt werden, dass der Beweiser das Geheimnis kennt.

In einem Kryptosystem in dem der öffentliche Schlüssel zwei isomorphe Graphen (G_0, G_1) sind und der private Schlüssel der Isomorphismus π zwischen den beiden. Der Beweiser P ist im Besitz von π und möchte einen Verifizierer V im Besitz des öffentlichen Schlüssels überzeugen, dass er π kennt.

1. P wählt ein zufälliges $a \in \{0, 1\}$ und eine zufällige Permutation ρ auf dem Graphen. Damit berechnet er $H := \rho(G_a)$ und sendet H an V (Commitment)
2. V wählt ein zufälliges $b \in \{0, 1\}$ und fordert P auf ihm eine Permutation σ mit $H = \sigma(G_b)$ zu senden (Challenge)
3. P wählt nun die Permutation wie folgt$\sigma := \begin{cases} \rho, & \text{falls } a = b \\ \rho \circ \pi^{-1}, & \text{falls } a = 0 \wedge b = 1 \\ \rho \circ \pi, & \text{falls } a = 1 \wedge b = 0 \end{cases}$ und sendet sie an V
4. V enpfängt σ und prüft ob $H = \sigma(G_b)$ gilt

P kann nur dann immer ein richtiges σ senden, wenn P im Besitz von π ist. Das Protokoll muss also hinreichend oft wiederholt werden um V wirklich zu überzeugen (Wahrscheinlichkeit von $1 - 2^{-n}$)

4 IT-Sicherheit

4.1 Zugriffskontrolle

Drei Modelle zur Zugriffskontrolle sind:

- Benutzerbestimmbare Zugriffkontrolle (DAC)
  - Eigentümer ist für die Zugriffberechtigungen zu Objekten selbst verantwortlich

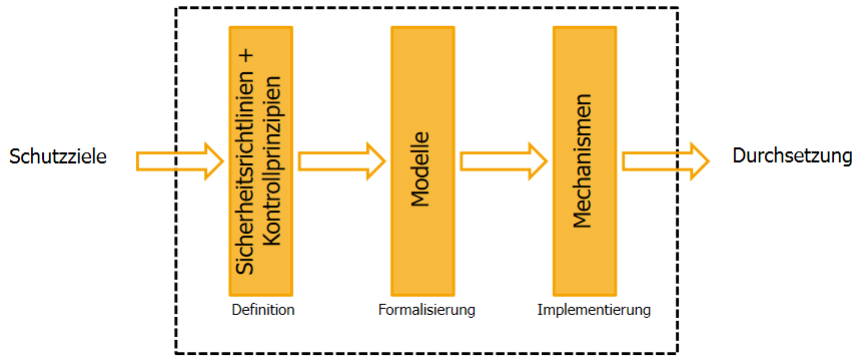


Abbildung 18: Kontrollprinzipien sollen die Einhaltung der Sicherheitsrichtlinien gewährleisten und werden durch Modelle abstrahiert. Mechanismen implementieren diese Modelle.

  - Rechte werden für einzelne Objekte vergeben
  - Objektbezogene Sicherheitseigenschaften, aber keine systemweiten
  - Fast alle Betriebssysteme unterstützen DAC
- Rollenbasierte Zugriffskontrolle (RBAC)
  - Berechtigungen für Rollen statt Benutzer (MS Active Directory)
- Systembestimmte Zugriffkontrolle (MAC)
  - Systembestimmte (regelbasierte) Festlegung von Sicherheitseigenschaften
  - Benutzerdefinierte Rechte werden durch systembestimmte überschrieben

4.1.1 Benutzerdefinierte Zugriffskontrolle

Dieses Modell ist einfach zu implementieren und intuitiv, flexibel einsetzbar.
Es gibt allerdings keine formale Garantien für den Informationsfluss, die Rechtevergabe/-rücknahme relativ komplex. Ein beschränkter Zugriff auf bestimmte Teile ist schwierig zu implementieren. Außerdem ist eine Rechtevergabe am Admin vorbei möglich.

Dargestellt wird ein DAC als eine Zugriffsmatrix. Formal geschrieben als:

Menge von Objekten:	O
Menge von Subjekten:	S
Menge von Rechten:	R z.B. $R = \{\text{read, write, execute, own}\}$
Matrix bestimmt Abbildung	$M : S \times O \to P(R)$

Da in einer Matrixrepräsentation viele Zellen leer bleiben wird eine Access Control List (ACL) verwendet um die Matrixeinträge zu speichern. In dieser stehen Tupel aus dem Prozess/Nutzer mit seinen entsprechenden Rechten.
Beispielsweise: $ACL(Datei1) = \{(Prozess1, \{read, write\}), (Prozess5, \{owner, execute\})\}$
Das ist eine objektbezogene Sicht der Zugriffsmatrix, eine andere Variante ist die subjektbezogene Sicht. Hier speichert man für alle Subjekte eine Liste von Objekten samt Zugriffsrechten.
Alle Rechte eines Nutzers sind effizient bestimmbar, aber die Rechte an einem spezifischen Objekt schwierig.
Beispielsweise: $CL(Prozess3) = \{(Datei2, \{owner, execute\}), (Prozess1, \{signal\})\}$

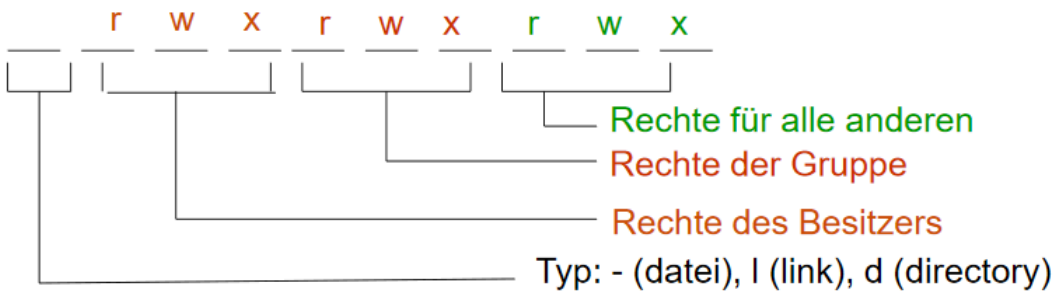


Abbildung 19: UNIX speichert die Rechte von Objekten in solchen Einträgen, okal kodiert. Jede Datei kann somit nur einen Besitzer haben

4.1.2 Rollenbasierte Zugriffskontrolle

Hier werden die Rechte an Gruppen vergeben denen Subjekte zugeordnet sind. Die Rollenmitgliedschaften sind im Vergleich zu DAC leicht zu verändern und unterstützt dynamischen und statischen Ausschluss, sowie Sitzungen (Sessions) sodass nicht alle Rollen gleichzeitig aktiv sind.
Formal schreibt man:

Menge von Subjekten:	S
Menge von Rollen:	R
Menge von Zugriffsechten:	P
Zuordnung Benutzer $\rightarrow$ Rollen:	$sr : S \rightarrow \mathcal{P}(R)$
Zuordnung Rolle $\rightarrow$ Rechte:	$pr : R \rightarrow \mathcal{P}(P)$

Eine Sitzung ist eine Relation $session \subseteq S \times \mathcal{P}(R)$ und modelliert die "aktive" Zuordnung von Subjekten auf Rollen. Ein Subjekt kann auch nicht aktive Rollen besitzen.
Mit der partiellen Ordnung $\leq \subseteq R \times R$ können wir Rollen nach ihren Rechten ordnen, sodass $R_i \leq R_j$ bedeutet, dass R_j alle Rechte von R_i hat und ggf. noch mehr.

Die Funktion $active : S \rightarrow \mathcal{P}(R)$ bildet auf alle aktiven und nach $\leq$ untergeordneten Rollen eines Subjekts ab.
Mit $member : R \rightarrow \mathcal{P}(S)$ bildet auf die Menge Subjekte an welche die gegebene Rolle hat.
Das Prädikat $exec : S \times P \rightarrow \mathbb{B}$ gibt an ob ein Subjekt aktuell eine Berechtigung hat.

In einer RBAC Implementierung muss immer gelten, dass ein Subjekt nur in den Rollen aktiv sein kann, in denen es Mitglied ist und darf nur die Rechte der aktiven Rollen besitzen.

Rollentrennung kann statisch mit $SSD \subseteq R \times R$ ausgeschlossen werden, definiert als:

$$\forall R_i, R_j \in R, \forall s \in S : (s \in member(R_i) \wedge s \in member(R_j)) \Rightarrow (R_i, R_j) \notin SSD$$

Dynamischen Ausschluss definiert man mit $DSD \subseteq R \times R$ wie folgt:

$$\forall R_i, R_j \in R, \forall s \in S :$$
$$(s \in member(R_i) \wedge s \in member(R_j) \wedge \{R_i, R_j\} \subseteq active(s)) \Rightarrow (R_i.R_j) \notin DSD$$

RBAC ist eine Erweiterung von DAC falls $pr = M$ gilt.

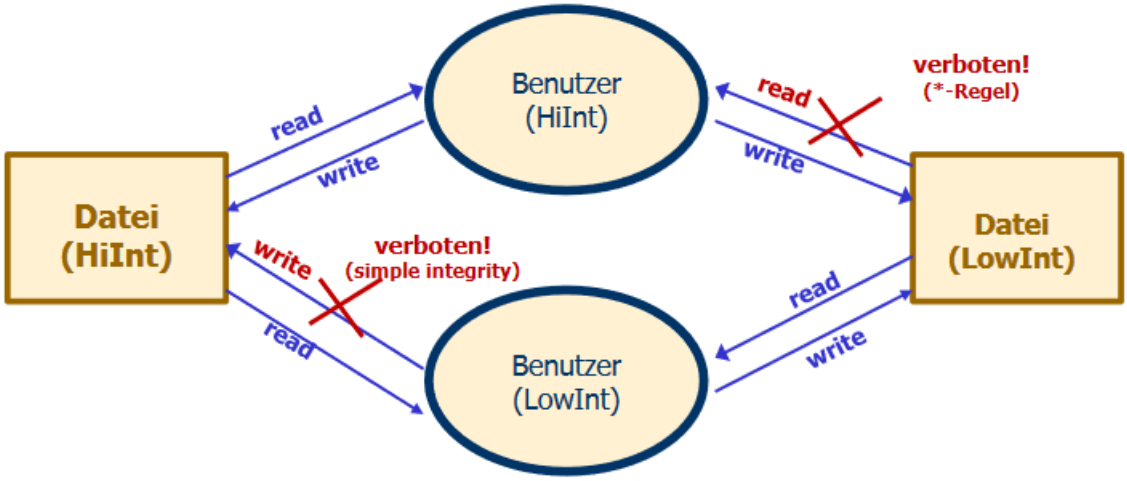


Abbildung 20: Als duale Modellzu Bell-La Padula sichert das Biba Modell die Integrität von Daten

4.1.3 Systembestimmte Zugriffskontrolle

Dieses Modell beschäftigt sich mit dem Informationsfluss mittels Zugriffsrechten, welche z.B. mit Multi-Level Security überwacht werden.

Das Bell-La Padula Modell erweitert die Zugriffsmatrix M um ein Zugriffskontrollsystem, indem es jedem Subjekt s und jedem Objekt o eine Sicherheitsmarkte (label) **L** ($vertraulich < geheim < streng\ geheim$) zuordnet.
Außerdem benötigen wir zum erstellen von Sicherheitsklassen **SC**$= L \times \mathcal{P}(C)$ noch Sicherheitskategorien **C** welche die Zuständigkeiten modellieren.
Jedem Subjekt wird eine Sicherheitsklasse (Ermächtigung) und jedem Objekt eine Sicherheitsklasse (Einstufung) zugewiesen.
Auf SC definieren wir eine partielle Ordnung $\leq$:
$\forall (l, c), (l', c') \in SC : (l, c) \leq (l', c') \Leftrightarrow l \leq l' \wedge c \subseteq c'$

Die Sicherheitseigenschaften von Mandatory Access Control sind:

- **Simple-security-Property** (no read up Regel)
  - Lesezugriff/Ausführung nur erlaubt falls $read \in M(s, o) \wedge SC(o) \leq SC(s)$
- ***-Property** (no write down-Regel)
  - Append nur erlaubt falls: $append \in M(s, o) \wedge SC(s) \leq SC(o)$
  - Read-Write nur erlaubt falls: $read - write \in M(s, o) \wedge SC(s) = SC(o)$
- **Strong Tranquility Regel**
  - Keine Änderung der Subjekt-Clearance oder der Objekt-Classification zur Systemlaufzeit

Nachteile des Bell-La Padula Modells sind, dass Daten tendenziell höher klassifiziert werden, Subjekte müssen möglicherweise blind schreiben (nur append). Außerdem können keine verdeckten Kanäle modelliert werden. Dafür ist das Modell einfach zu implementieren und bildet herarchische Informationsflüsse nach.

4.2 Authentifizierung

Wir definieren folgende Begriffe:

- Identität, als eine Menge von Attributen
- Authentisierung: Bereitstellung von Unterlagen, die es ermöglichen, die Identität zu prüfen
- Authentifizierung: Prüfung und Echtheitsbezeugung der Unterlagen zu Identitätsfeststellung
- Autorisierung: Gewährung/Verwehrung von Rechten

Ansätze zur Authentifizierung sind:

- Authentifizierung durch Wissen (z.B. PIN, Passwort, krypt. Schlüssel)
- Authentifizierung durch Besitz (z.B. Smartcard, Token, SIM-Karte)
- Authentisierung durch Merkmale (z.B. Biometrie)

Verschiedene Ansätze werden normalerweise zu einem Mehrfaktorverfahren kombiniert.
Bei der **einseitigen Authentifizierung** authentisiert sich eine Partei A gegenüber einer anderen Partei B (B authentifiziert A). Zum Beispiel Authentifiziert sich ein Nutzer gegenüber eines PC.
Bei der **wechselseitigen Authentifizierung** authentifizieren sich zwei Parteien gegenseitig. Zum Beispiel Online-Banking bei Verwendung von Zertifikaten.

4.2.1 Authentisierung durch Wissen

Die gängigste Methode sind Passwörter, welche üblicherweise als Hashwert gespeichert und durch Salts gegen Offline-Attacken weiter geschützt werden können.
Probleme sind:

- Schwache Passwörter ("dictionary attack")
- Benutzer verwenden gleiches Passwort bei verschiedenen Servern
- Manche UNIX-Tools (telnet, ftp) übertragen Passwörter im Klartext
- NIS, YP übertragen Hashes ("Offline Attacke")

Besser wäre ein Authentifizierungsverfahren welches die nicht Übertragung eines Passwortes erfordern würde.
Arten Passwörter zu übertragen:

- Plaintext: Ein passiver Angreifer kann alles mitlesen
- Plaintext, TLS geschützt: Angreifer auf Datenbank erlangt alle Passwörter
- Hash, TLS geschützt: Angriff mit Rainbowtable, da bei eindeutigen Hashalgo. das selbe Passwort für alle Nutzer den selben Hash erzeugt
- Hash mit Salt, TLS geschützt: Angriff mit Rainbowtable gibt keinen Geschwindigkeitsgewinn, da der Hash nun nicht mehr allein vom Passwort abhängt.

Authentifizierung durch Wissen kann auch mit Challenge-Response-Verfahren (CHAP) funktionieren.

1. Server sendet Challenge c
2. Nutzer verwendet Integritätsschutzverfahren auf Nachricht $m := c$

3. Server prüft ob Nachricht $m = c$ und prüft Authentizität via Verfahren

Wie hier zu sehen ist, muss Bob ebenfalls das Passwort speichern um c zu verifizieren. Auch wenn $K_{ID} = h(passwort)$ gesetzt wird braucht ein Angreifer in diesem Fall nur K_{ID}. Auch muss der Klartextraum von RAND groß sein, damit ein Angreifer keine bereits benutzte Challenge wiederverwenden kann (Replay-Attack).
Das Verfahren beweist Bob nur dass Alice sich authentisieren möchte, wenn authentifizierte Daten übertragen werden sollen, muss die Integrität des Kommunikationskanals geschutzt werden (Man-in-the-middle Angriff).

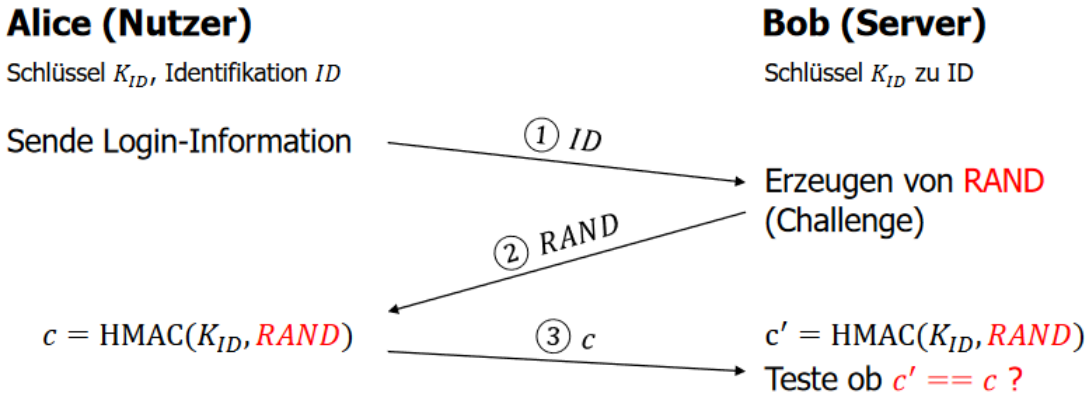


Abbildung 21: Beide Parteien kennen den gemeinsamen Schlüssel K_{ID}, anstelle von HMAC kann auch eine Verschlüsselungsfunktion verwendet werden

4.2.2 Authentisierung durch Besitz

Nutzer authentifizieren sich mit Tokens welche statisch - ein gespeichertes Geheimnis (Passwort), oder dynamisch - Geheimnis wird zur Berechnung von Authentifizierungs-Informationen genutzt (z.B. CHAP mit SmartCard mit Kryptoprozessor).
Token können als Hardware (Schlüssel, SmartCard, Transponder) oder Software (Cookie, Client-Zertifikat) vorliegen.

Probleme sind

- Diebstahl
  - Nutzung des Tokens durch zusätzliche Authentifikation sichern
    - Software-Crypto: Privaten Schlüssel mit Passwort verschlüsseln
    - Hardware-Crypto: Token mit Pin
  - Nutzung mit mit zusätzlichem zweiten Faktor
- Extraktion der kryptographischen Schlüssel (Kopie)
  - Schwachstellen in Tokenfirmware
  - Side-Channel Angriffe

4.2.3 Authentisierung durch Merkmale

Anforderungen:

- **Universalität:** Jede Person besitzt das Merkmal

- **Eindeutigkeit:** Merkmal ist für jede Person verschieden
- **Beständigkeit:** Merkmal ist unveränderlich
- quantitative **Erfassbarkeit** mittels Sensoren
- **Performanz:** Genauigkeit und Geschwindigkeit
- **Akzeptanz** des Merkmals beim Benutzer
- **Fälschungssicherheit**

Die Erkennung läuft in zwei Prozessen, dem **Enrollment**, d.h. der Registrierung eines Benutzers, und der **Verifikation**, d.h. der Erfassung und dem Vergleich mit im Enrollment erfassten Daten.

Die Verifikation ist immer fehlerbehaftet:

- False positives: Unberechtigter wird authentifiziert (FAR) → Sicherheitsproblem
- False negatives: Berechtigter wird abgewiesen (FRR) → Probleme bei Akzeptanz, Benutzbarkeit
- FAR und FRR können durch Hinzunahme von Merkmalen gesteuert werden
- Equal Error Rate, wenn Anzahl an FAR und FRR gleich sind
- Meist ungeprüfte Hersteller-Angaben

Einzelne feine Punkte in Fingerabdrücken bezeichnet man als Minutien. Gemeint sind Verzweigungen, Endpunkte, etc. jeweils mit (relativen) Koordinaten und Winkeln
Deren Muster scheint eindeutig zu sein. Minutien können allerdings durch schlechte Abdrücke fehlen. Bei der Verifikation werden vorhandene Minutien mit jenen die beim Enrollment extrahiert wurden verglichen.
Biometrie birgt Datenschutzprobleme, da es Rückschlüsse auf medizinische Daten (Venenmuster) und Gesundheitszustand ermöglicht. Anstelle von Merkmalen sollten lediglich Referenzdaten gespeichert werden aus denen keine Rekonstruktion möglich ist.

Sensoren können durch künstliche Fingerabrücke, Klone durch hochauflösende Fotos oder Gewaltkriminalität (abgeschnittener Finger) getäuscht werden.
Biometriedaten sind nicht (DNA) bzw. nur begrenzt (Fingerabdruck, Iris) widerrufbar. Wenn Kopien prinzipiell erstellt werden können, stellt die Kompromittierung eines Lesegerätes die Authentifizierung aller anderen in Frage.

4.2.4 Kerberos

Ziel von Kerberos ist die zentrale Authentifizierung von Subjekten (Principals), sodass keine separate Authentisierung bei Dienstnutzung erforderlich ist (Single-Sign-on).
Jede Domäne besitzt ein Key Distribution Center (KDC) welches aus einem Authentification Server (AS), welcher Principals seiner Domäne authentifiziert, und einem Ticket Granting Server (TGS), welcher Tickets als Identitätsausweis ausstellt.
Basis der Authentifizierung sind Pre-Shared Secrets zwischen KDC und Principal, d.h. das KDC kennt die Secrets aller Netzwerkteilnehmer.

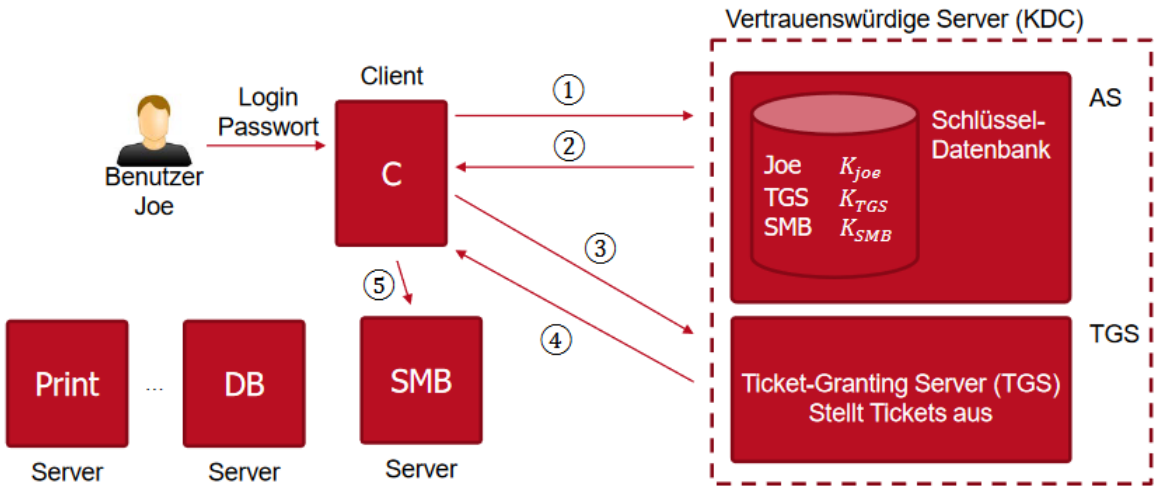


Abbildung 22: Loginverfahren bei Kerberos. Nummerierung wie hier

Ein vom AS ausgestelltes Ticket Granting Ticket (TGT) beinhaltet:
$T_{C,S} = (S, C, addr, timestamp, lifetime, K_{C,S})$

S	Name des Servers
C	Name des anfordernden Clients
$addr$	IP-Adresse des Clients
$timestamp$	aktuelle Zeit
$lifetime$	Lebenszeit des Tickets
$K_{C,S}$	Sitzungs-Schlüssel für Kommunikation zwischen C und S

Der Login funktioniert wie folgt:
Joe gibt Passwort auf lokalem Client C ein, $K_{joe} := Hash(Passwort)$ wird generiert.

(1) $$Joe \rightarrow AS : \{timestamp\}_{K_{joe}}, Joe, N_1, TGS$$
(2) $$AS \rightarrow Joe : \{K_{joe,TGS}\}_{K_{joe}}, \{TGT\}_{K_{TGS}}$$
(3) $$Joe \rightarrow TGS : \{A_{joe}\}_{K_{joe,TGS}}, \{TGT\}_{K_{TGS}}, SMB, N_2$$
(4) $$TGS \rightarrow C : \{K_{joe,SMB}, N_2\}_{K_{joe,TGS}}, \{T_{joe,SMB}\}_{K_{SMB}}$$
(5) $$C \rightarrow SMB : \{A_{joe}\}_{K_{joe,SMB}}, \{T_{joe,SMB}\}_{K_{SMB}}$$
(6) $$SMB \rightarrow C : \{timestamp + 1\}_{K_{joe,SMB}}$$

Hierbei sind N_1, N_2 Nonces und $A_{joe} = (Joe, IP - Addr, timestamp)$ der Authenticator ist. Der sechste Schritt ist optional falls wechselseitige Authentifizierung gewünscht ist. Wenn der AS $\{timestamp\}_{K_{joe}}$ entschlüsseln kann und der Zeitstempel in den letzten 5 Minuten liegt, wird ein Ticket-Granting-Ticket TGT (beinhaltet $K_{joe,TGS}$) ausgestellt. Joe kann nach Empfangen von (2) den Sitzungsschlüssel mit TGS entschlüsseln und den zweiten Teil in (3) an TGS senden sodass TGS sicher mit Joe kommunizieren kann. Außerdem teilt Joe TGS mit, für welchen Dienst er ein Ticket möchte.

Wenn TGS den Authenticator erfolgreich überprüft hat, sendet er in (4) einen Sitzungsschlüssel $K_{joe,SMB}$ für Kommunikation zwischen Joe und SMB, sowie ein Ticket für SMB. Joe kann nun das Ticket beim SMB-Server benutzen.

Angriffe auf Kerberos:

Beim Offline Cracking wird Übertragung (1) abgefangen und $\{timestamp\}_{K_{joe}}$ versucht zu entschlüsseln. Sollte das Ergebnis des Entschlüsselns die Form eines Zeitstempels haben, wird eine Checksum erzeugt und mit der des tatsächlichen Pakets verglichen. Stimme beide überein ist das Passwort gefunden, sonst wird das nächste ausprobiert.

Overpass-the-hash, hier kann ein Angreifer sich gegenüber Kerberos als ein bestimmter Nutzer ausgeben, falls er den Hash eines Nutzers kennt (K_{joe} kann auch der Hash des Passwortes sein, wie hier).

Pass-the-Ticket, hier kennt der Angreifer ein Ticket des Nutzers und nutzt dieses um im Namen des Nutzers auf Dienste zuzugreifen. Angriff beginnt ab (3)

Golden Ticket, hier stellt der Angreifer sich selbst ein TGT aus, wobei er den Masterkey des Domain Controllers besitzt. Dadurch kann er in (3) ein sein TGT einfügen und kann so vom TGS Zugriff auf jeden Dienst anfordern.

Silver Ticket, der Angreifer kennt den Hash/Key eines Application Servers und kann damit im Namen des Servers Zugriff auf alle Dienste anfordern, auf die auch der Server zugriff hat. Dabei tritt er nicht mit dem Domain Controller in Kontakt