Computer System Sicherheit
Notizen WS20/21

Felix Marx

2. Dezember 2020

Inhaltsverzeichnis

1 Begriffe **2**
- 1.1 Security 2
- 1.2 Safety 3

2 Verlässliche Systeme **3**
- 2.1 Strategien zur Fehlervermeidung/toleranz 4

3 Krypographie **6**
- 3.1 Begriffe 6
- 3.2 Verschiebechiffre 7
- 3.3 Mathematische Grundlagen 7
  - 3.3.1 Teilbarkeitsregeln 7
  - 3.3.2 Größter gemeinsamer Teiler 7
  - 3.3.3 Primzahlen 8
- 3.4 Symmetrische Kryptosysteme 8
  - 3.4.1 Vigenère mit alphabetischen Schlüssel, Länge n 9
  - 3.4.2 Electronic Codebook Modus 9
  - 3.4.3 Cipher Block Chaining Modus 10
  - 3.4.4 Stromchiffren 11
- 3.5 Asymmetrische Kryptosysteme 12
  - 3.5.1 RSA-Kryptosystem 12
  - 3.5.2 Elgramal-Kryptosystem 13
- 3.6 Hashfunktionen 13
  - 3.6.1 Merkle-Damgård-Konstruktion 14
  - 3.6.2 Exkurs: Chosen Target Forced Prefix 14
  - 3.6.3 HMAC-Konstruktion 16

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"

Herausforderungen:

- **Vertraulichkeit** der Daten: „drahtloser Informationsdiebstahl", Verlust des Geräts
- **Integrität** der Daten: Zugriffskontrolle, Schutz vor Malware?
- **Zusammenspiel mit anderen Geräten**: Denial of Service?

Abbildung 1: Herausforderungen bei der Mobilität

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

Herausforderungen:

Vertrauenswürdige Infrastrukturen: Einfluss von Programmier- bzw. Designfehlern, aktive Angriffe, Heterogenität
Skalierung: Performance, Verhindern von DDoS
Datenschutz („Privacy")

Abbildung 2: Herausforderungen bei der Vernetzung

Miniaturisierung, Produktion kleinerer Chips mit Trend zu Einwegchips

Herausforderungen:

Ressourcen: limitierte Rechenleistung, Sicherheitskomponenten müssen kostengünstig sein
Privacy: Tracing, Identifikation von Personen

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.

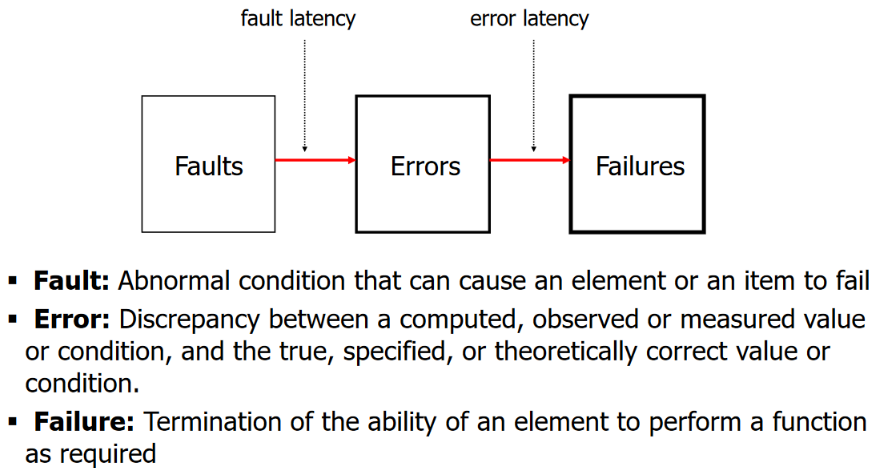

Abbildung 4: Ablauf bis ein System versagt

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:

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

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

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

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

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

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

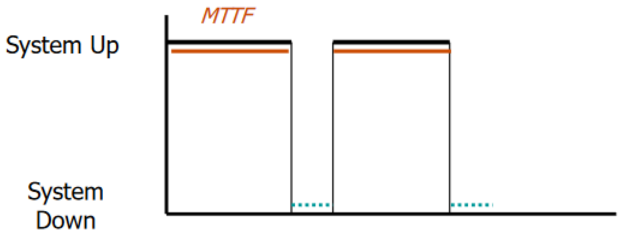

Abbildung 5: Berechnung MTTF mit dem exponentiellen Model

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)
  - Schutz vor Fälschung von Informationen / Daten
- 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 bei denen es "einfach" ist den Funktionswert $y = f(x) \in Y$ zu bestimmen, aber "schwer" ein Urbild $x \in f^{-1}[\{y\}]$ zu finden. Die Existenz von Einwegfunktionen ist nicht bewiesen (P/NP).

3.3.1 Teilbarkeitsregeln

$$a|a \tag{1}$$
$$a|b \wedge b|c \Rightarrow a|c \tag{2}$$
$$a|b \wedge b \neq 0 \Rightarrow |a| \leq |b| \tag{3}$$
$$a|b \wedge b|a \Rightarrow |a| = |b| \tag{4}$$

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

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

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

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

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

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 $m^{\varphi(n)} \mathrm{mod}\ n = 1$.
Kleiner Satz von Fermat:
Für eine Primzahl p und ein teilerfremde Zahl $m \in \mathbb{N}$ gilt $m^{p-1} \mathrm{mod}\ p = 1$.

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 $pad : M^* \rightarrow (M^n)^*$ 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.

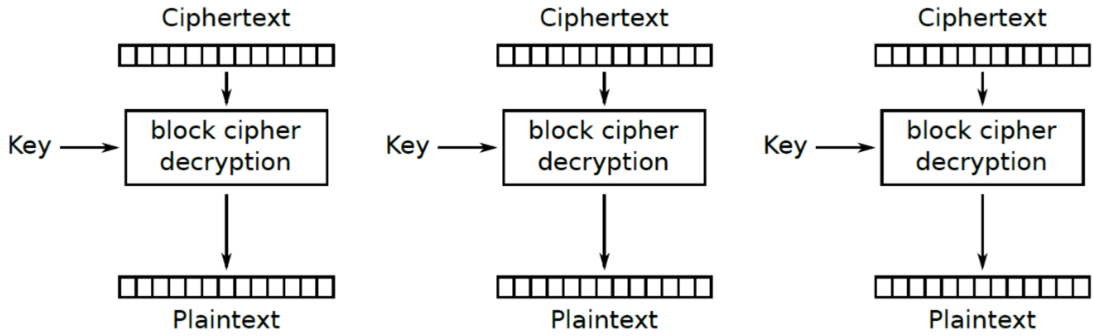

Abbildung 7: Entschlüsselung im ECB Modus

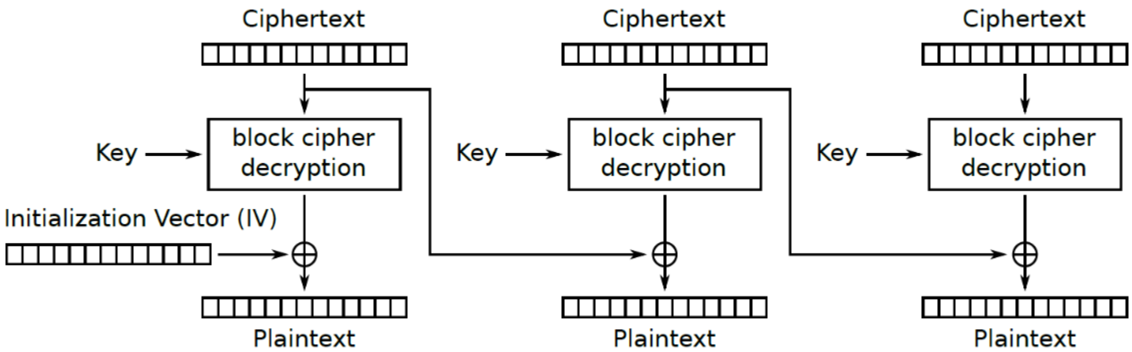

Abbildung 8: Entschlüsselung im CBC Modus, bei der Entschlüsselung umgedreht, sodass Parallelisierung unmöglich ist

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

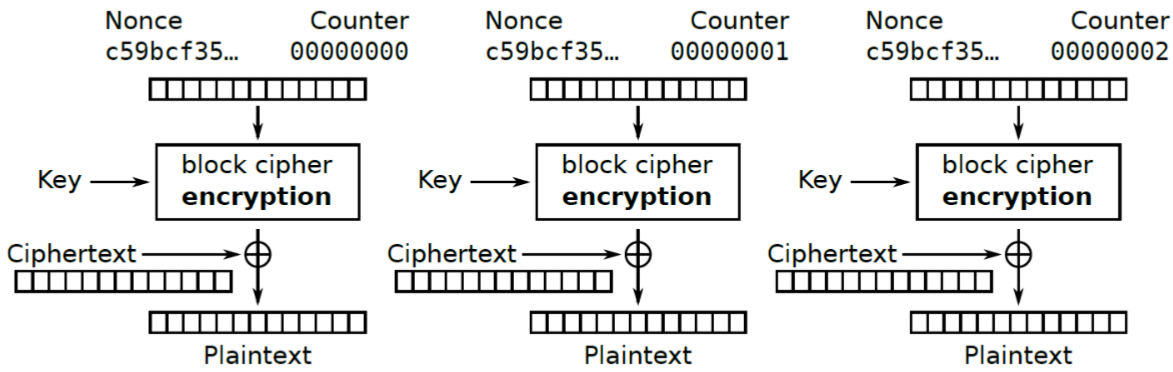

Abbildung 9: Entschlüsselung im CTR Modus, Nonce ist der Initialiserungsvektor

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 $1 \neq e \in \mathbb{Z}_{\varphi(n)}^{\times}$ so dass $\mathrm{ggT}(e, \varphi(n)) = 1$ gilt
4. Berechne den Entschlüsselungsexponenten $d \in \mathbb{Z}_{\varphi(n)}^{\times}$, 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 Elgramal-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

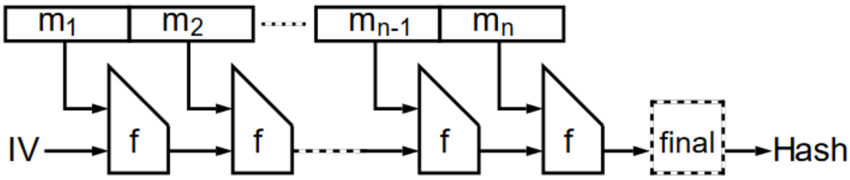

Abbildung 10: Bei Merkle-Damgård wird jeder Block mit den vorigen Block komprimiert und dann mit g finalisiert

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[6]
- (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, $f : A^{n+m} \rightarrow A^n$ eine Kompressionsfunktion,
pad:$A^* \rightarrow (A^m)^*$ eine Auffüllfunktion, $x_0 \in A^n$ ein beliebiger Initialisierungsvektor und g:$A^n \rightarrow A^n$ eine Finalisierungsfunktion.
Dann ist die Hashfunktion $h : A^* \rightarrow A^n$ für $x \in A^*$ definiert durch

1. $x_1x_2...x_k = pad(x)$ 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[7] konstruieren:

[6]Eine Funktion deren Funktionswert $y = f(x)$ zu einem gegebenen x in polynomieller Zeit berechenbar ist, aber das Urbild x zu einem gegebenen y nicht in Polynomialzeit berechenbar ist

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

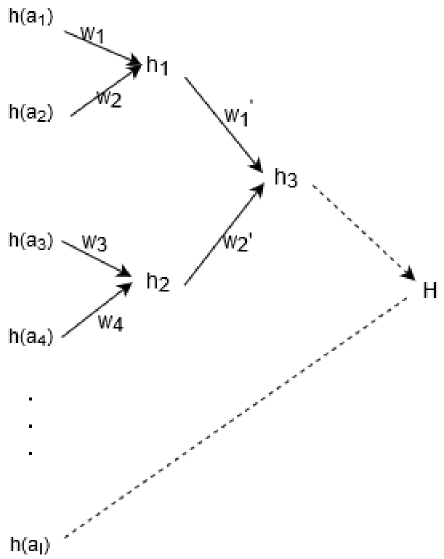

Abbildung 11: Dieser Hashbaum gibt Pfade an um aus 2^k Hashes den gewünschten Hash durch Konkatenation der Strings zu erreichen.

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

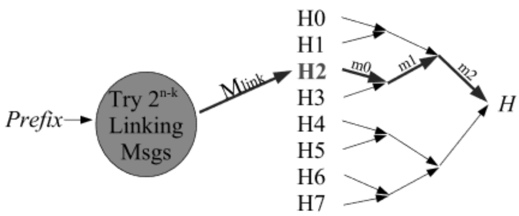

Abbildung 12: Nachdem ein Linkender String gefunden wurde, kann die gefälschte Nachricht zusammengesetzt werden

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 $H : \mathbb{Z}_2^* \rightarrow \mathbb{Z}_2^{8n}$ 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 $HMAC : \mathbb{Z}_2^* \times \mathbb{Z}_2^* \rightarrow \mathbb{Z}_2^*$ 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.