

# **Kryptographie im Studiengang SMK**

Matthias Vögeler

SoSe 2019

## **Inhaltsverzeichnis**

<b>1 Authentifizierung mit Hilfe von Passwörtern</b>	<b>5</b>
<b>2 Hash-Funktionen</b>	<b>7</b>
<b>3 Passwort-Cracking</b>	<b>12</b>
<b>4 Hash-Pointer und Datenstrukturen</b>	<b>20</b>
<b>5 Nachrichtenauthentifizierung</b>	<b>27</b>
<b>6 Wechselcodeverfahren</b>	<b>29</b>
<b>7 Historische Verschlüsselungsverfahren</b>	<b>32</b>
<b>8 Verschlüsselung durch Blockchiffren</b>	<b>42</b>
<b>9 Verschlüsselung durch Stromchiffren</b>	<b>55</b>
<b>10 Modulare Arithmetik</b>	<b>57</b>
<b>11 Public-Key-Kryptographie</b>	<b>60</b>
<b>12 Der RSA-Algorithmus</b>	<b>63</b>
<b>13 Diffie-Hellman Schlüsselaustauschverfahren</b>	<b>67</b>
<b>14 Elliptische Kurven Kryptographie</b>	<b>71</b>

## Abbildungsverzeichnis

1	Quelle: Matt Crypto	9
2	Quelle: Matt Crypto	10
3	Knackzeit durch Brute Force. Quelle: c't 3/2013	12
4	Regal mit Grafikkarten	12
5	Häufige Passwörter und PINs	13
6	10,000 Top Passwords	13
7	Quelle: Wikipedia	14
8	Quelle: Wikipedia	14
9	Quelle: xkcd	16
10	Quelle: Apple	16
11	Quelle: c't 18, 2014	17
12	Quelle: c't 23/2016	18
13	Quelle: c't 23/2016	18
14	Quelle: Bitcoin and Cryptocurrency Technologies	20
15	Quelle: Bitcoin and Cryptocurrency Technologies	21
16	Quelle: Bitcoin and Cryptocurrency Technologies	21
17	Vereinfachte Bitcoin Blockchain	22
18	Anzahl der Bitcoin-Transaktionen pro Tag	23
19	Wachstum der Blockchain über die Jahre	24
20	Evolution des Schürfens	25
21	Bitcoins und Internetkriminalität	26
22	Merkle-Damgård-Konstruktion	28
23	Yubikey	31
24	Quelle: Wikipedia	32
25	Quelle: WDR	33
26	Vigenère-Quadrat	34
27	Quelle: Beutelspacher et al.: Kryptografie in Theorie und Praxis	40
28	Quelle: Ertel: Angewandte Kryptographie, Hanser	41
29	Vergleich der AES-Kandidaten	46
30	Rijndael / AES – Ablauf einer Runde	48
31	In jeder Runde wird der aktuelle Zustand mit dem Rundenschlüssel XORed.	49
32	Zeilen der Zustandsmatrix werden nach links rotiert.	50
33	Quelle: Moserware	51
34	Electronic-Codebook-Modus (ECB)	51
35	Quelle: StackExchange – Why shouldn't I use ECB encryption?	52
36	Cipher-Block-Chaining-Modus (CBC)	52
37	Cipher-Feedback-Modus (CFB)	53
38	Counter-Modus (CTR)	53
39	Funktionsprinzip einer Stromchiffre	55
40	Das Schieberegister $u_i = u_{i-1} \oplus u_{i-6} \oplus u_{i-8}$	55
41	A5-1 GSM-Stromchiffre	56
42	Quelle: Toolsley	60

43	Quelle: Toolsley . . . . .	61
44	RSA mit geteiltem Modul . . . . .	65
45	Exponentialzyklen . . . . .	68
46	Zwei Beispielkurven . . . . .	71
47	Graphische Darstellung der Punktaddition . . . . .	72
48	Projektive graphische Darstellung der Punktaddition . . . . .	72
49	Diskrete elliptische Kurve . . . . .	73
50	Berechnung von Bitcoin-Adressen . . . . .	74
51	Ken Shirriff's blog: Bitcoins the hard way . . . . .	76

# 1 Authentifizierung mit Hilfe von Passwörtern

## Das Passwort

- Das Passwort nennt man auch: Kennwort, Schlüsselwort, Codewort, Losung, Parole, PIN, Shared Secret
- Passwörter dienen der Authentifizierung → wichtiges Schutzziel
- Es wird eine Zeichenfolge vereinbart, die zum Ausweisen für einen Zugang benötigt wird.
- Die Authentizität des sich so Ausweisenden hängt unmittelbar von der Geheimhaltung des Passwortes ab.

## Einsatz von Passwörtern

- Parole beim Militär,
- im Internet in Kombination mit einem Benutzerkonto
- als gemeinsames Geheimnis (Shared Secret) mehrerer Kommunikationspartner
- Einmalkennwörter beim TAN-Verfahren
- (Zero protection from nuclear code)

## Die Sicherheit von Passwörtern

### Bedrohung von Passwörtern im Internet

- vorhersehbar oder zu schwach,
- kann abgefangen werden,
- Phishing

### Die Länge von Passwörtern

Passwörter können erraten oder durchprobiert werden. Die Mindestlänge eines Passwörter, um der Bedrohung durch Erraten/Durchprobieren entgegen zu wirken, hängt von seinem Einsatz ab.

Das BSI empfiehlt,

- für Internetzugänge Passwörter mit mindestens 12 Groß- und Kleinbuchstaben sowie Sonderzeichen und Ziffern zu verwenden,
- für WLAN-Zugänge hingegen Passwörter aus mindestens 20 Zeichen.

Hingegen besteht eine EC-Karten PIN nur aus 4 Ziffern.

### Die Länge von Passwörtern

Was ist besser, mehr Länge oder mehr Auswahl? Was ist besser bei einer Länge von 8 auf 9 zu gehen oder die Anzahl der Symbole von 26 auf 36 zu erhöhen?

Für die Anzahl  $N$  möglicher Passwörter der Länge  $n$  eines Alphabets mit  $b$  Symbolen gilt

$$N = b^n$$

Wie groß ist  $N$  bei einem Passwort der Länge 12, wenn das Alphabet die Buchstaben a-z, A-Z, 0-9 und die Sonderzeichen ! ? \$ % & / () [] {} = + - \_ enthält?

$$N = 78^{12} \approx 5 \cdot 10^{22}$$

Testet man 1 Mrd. Schlüssel pro Sekunde, ist man in 3 Millionen Jahren durch.

Wie schnell ein Angriff auf Passwörter mittels Durchprobieren durchgeführt werden kann, hängt entscheidend vom Umfeld ab:

- **Online:** 100-1000 Versuche pro Sekunde
- **Offline:** bis zu mehreren Milliarden Versuche pro Sekunde.

Welche Gegenmaßnahme fallen Ihnen ein, um das Durchprobieren von Passwörtern zu erschweren?

Oft werden Passworddateien direkt von einem Server geladen. Kennen Sie eine Methode, damit das Speichern im Klartext nicht notwendig ist?

**Hash-Funktionen**

## 2 Hash-Funktionen

### Einwegfunktionen

Eine Hash-Funktion  $h(x)$  bildet beliebig lange Zeichenketten  $x$  auf eine Zeichenkette fester Länge ab.

Folgende Eigenschaften sind dabei wichtig:

1.  $f$  ist eine Einwegfunktion, d.h. sie ist leicht berechenbar, aber praktisch unmöglich umzukehren.
2. Schwache Kollisionsresistenz: Zu einem gegebenen Hash-Wert  $z = h(x)$  ist es praktisch unmöglich ein  $x' \neq x$  zu finden, sodass  $h(x') = z$ .
3. Starke Kollisionsresistenz: Es ist praktisch unmöglich zwei verschiedene Zeichenketten  $x$  und  $x'$  zu finden, so dass  $h(x) = h(x')$ .

### Angriffe auf Kollisionsresistenz

Angenommen der stärkste Angriff auf die Kollisionsresistenz einer Hash-Funktion mit der Ausgabe von  $n$ -Bit basiert auf Raten (Brute-Force).

#### Schwache Kollisionsresistenz

Sei  $z = h(x)$  gegeben. Dann gibt es  $2^n$  mögliche Ausgaben. Dann benötigt man

$$\frac{2^n}{2} = 2^{n-1}$$

Versuche, um eine Kollision mit einer Wahrscheinlichkeit von 50% durch Raten zu finden, d.h. ein  $x' \neq x$  zu finden mit  $z = h(x')$ .

#### Starke Kollisionsresistenz

Wie viele Hash-Werte  $z_i$ ,  $i = 1, \dots, k$ , muss man berechnen, damit die Wahrscheinlichkeit eine Kollision zu erhalten 50% beträgt?

$$\begin{aligned} P(\text{Mindestens zwei der } z_i \text{ sind gleich.}) &= 1 - P(\text{Alle der } z_i \text{ sind verschieden.}) \\ &= 1 - \prod_{i=1}^{k-1} \frac{2^n - i}{2^n} = 1 - \prod_{i=1}^{k-1} \left(1 - \frac{i}{2^n}\right) \\ &\stackrel{k \ll 2^n}{\approx} 1 - \prod_{i=1}^{k-1} e^{-\frac{i}{2^n}} = 1 - e^{-\frac{1+2+\dots+k-1}{2^n}} && \left(e^x = 1 + x + \frac{x^2}{2!} + \dots\right) \\ &= 1 - e^{-\frac{k(k-1)}{2 \cdot 2^n}} \stackrel{!}{=} \frac{1}{2} && (\text{Gaußsche-Summenformel}) \end{aligned}$$

Mit  $k^2 \approx k(k-1)$  erhält man  $k \approx 2^{\frac{n+1}{2}} \sqrt{-\ln(1/2)} \approx 2^{\frac{n}{2}}$ .

## In der Praxis verwendete Einwegfunktionen

- **Message-Digest Algorithm 5 (MD5):** 1991, Ronald L. Rivest Generiert aus einer beliebigen Nachricht einen 128-Bit Wert. Kollisionen gefunden. Ist unsicher für kryptographische Zwecke.
- **Secure Hash Algorithm (SHA-1):** 1995, NIST & NSA Generiert aus einer beliebigen Nachricht einen 160-Bit Wert. Erste [Kollision](#) gefunden. Ist unsicher für kryptographische Zwecke.
- **Secure Hash Algorithm (SHA-2):** 2001, NIST Generiert aus einer beliebigen Nachricht einen 224/256/384/512-Bit Wert. Das BSI/NIST empfiehlt den SHA-2 zu verwenden.
- **Secure Hash Algorithm (SHA-3):** 2011, Bertoni, Daemen, Peeters, Van Assche: Gewinner des NIST-Wettbewerbs (2012) unter dem Namen Keccak Generiert aus einer beliebigen Nachricht einen 224/256/384/512-Bit Wert. Sicherheit noch unklar.
- **Race Integrity Primitives Evaluation Message Digest (RIPEMD-160):** 1996, Dobbertin, Bosselaers, Preneel Generiert aus einer beliebigen Nachricht einen 160-Bit Wert. Gilt als sicher. Wenig untersucht.
- **Whirlpool:** 2003, Rijmen und Barreto Generiert aus einer Nachricht ( $\leq 2^{256}$  Bit) einen 512-Bit Wert. Gilt als sicher. Wenig untersucht. Patentfrei
- **scrypt:** 2010, Percival Spezielles rechenintensives Passwort-Hashing-Verfahren. Schwierig auf spezieller Hardware zu implementieren.

## Message-Digest Algorithm 5

Hash-Algorithmen sind so konstruiert, dass eine Änderung nur eines Zeichens, d.h. eines Eingabebits, im Mittel die Änderung von der Hälfte des Ausgabebits zur Folge hat. D.h. jedes Ausgabebit ändert sich mit der Wahrscheinlichkeit 1/2:

```
$ echo -n "Frank mag Pizza" | md5sum  
0a3bf4fb3983547cd72413cf8dc4d162 -  
$ echo -n "Franz mag Pizza" | md5sum  
5070c56643ac6e6ba46c8b9ba93d439f -
```

## Message-Digest Algorithm 5

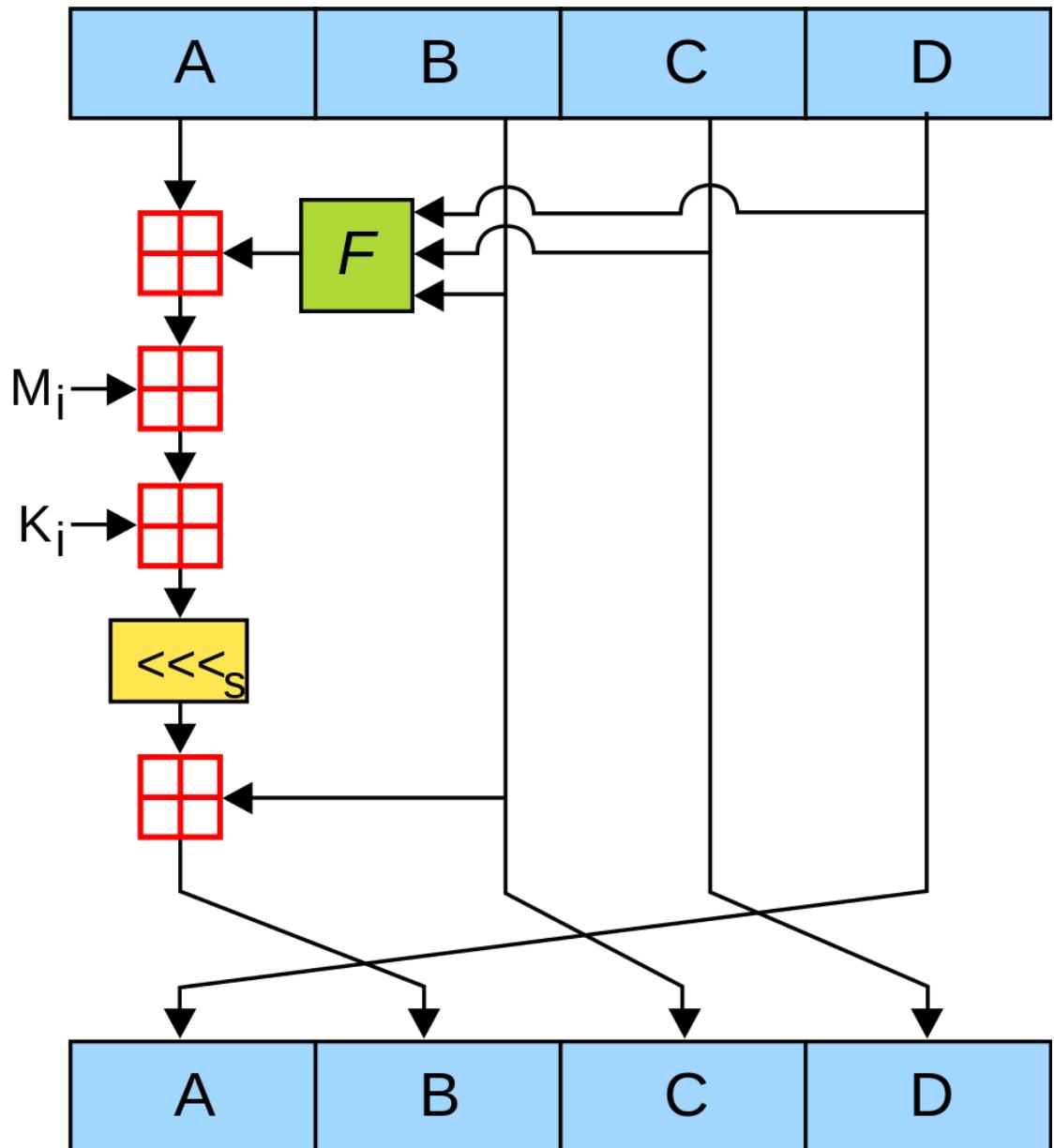


Abbildung 1: Quelle: Matt Crypto

Eine MD5-Operation besteht aus 64 Operationen dieses Typs, gruppiert in 4 Durchläufen mit jeweils 16 Operationen. A,B,C,D sind 32-Bit-Register, die zu Beginn mit Konstanten geladen werden.  $F$  ist eine nichtlineare Funktion, die im jeweiligen Durchlauf eingesetzt wird.  $M_i$  bezeichnet einen 32-Bit-Block des Eingabestroms und  $K_i$  eine für jede Operation unterschiedliche 32-Bit-Konstante.  $\lll_s$  bezeichnet die bitweise Linksrotation um  $s$  Stellen, wobei  $s$  für jede Operation variiert.  $\boxplus$  bezeichnet die Addition modulo  $2^{32}$ .

### Secure Hash Algorithm (SHA-1)

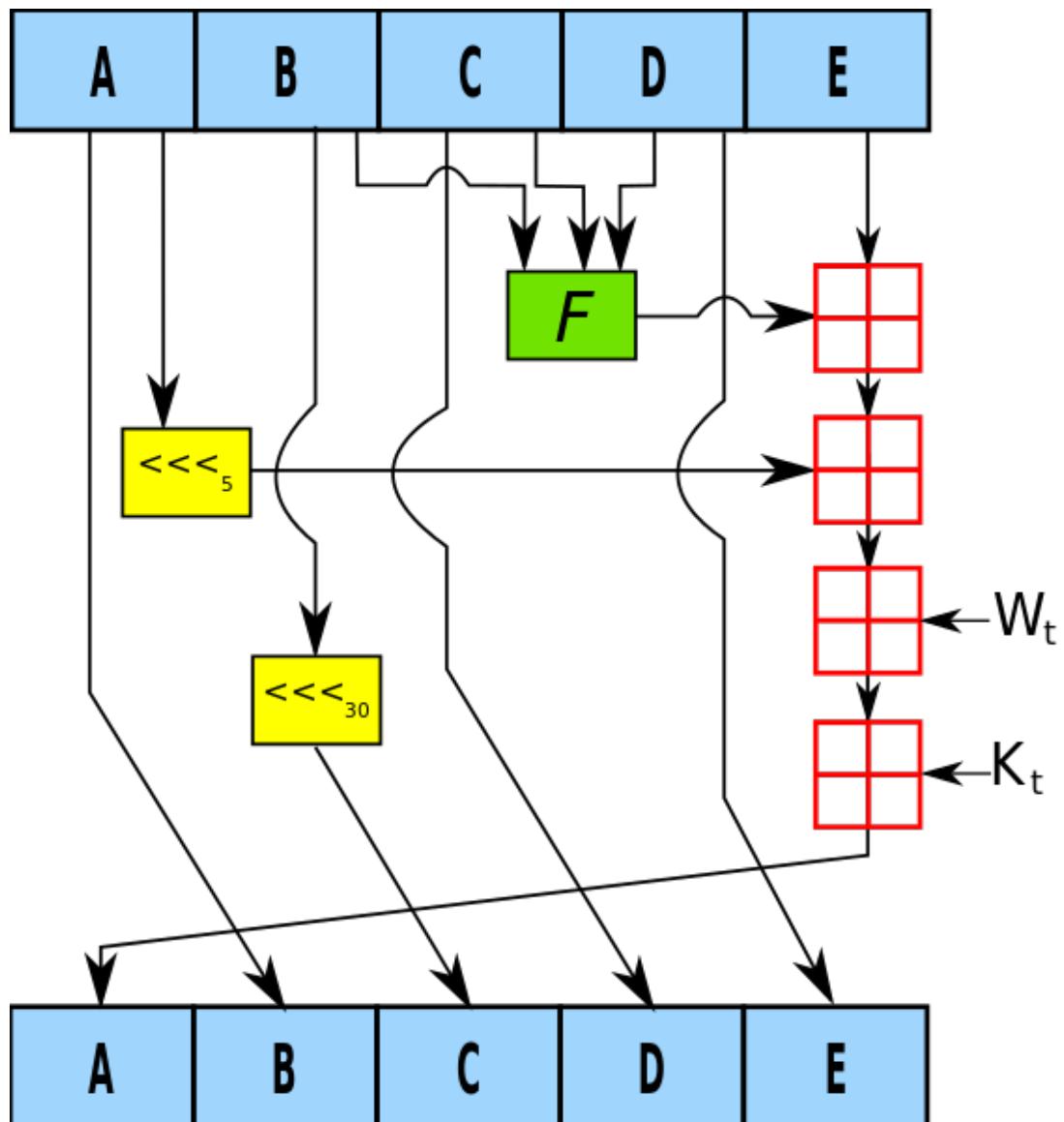


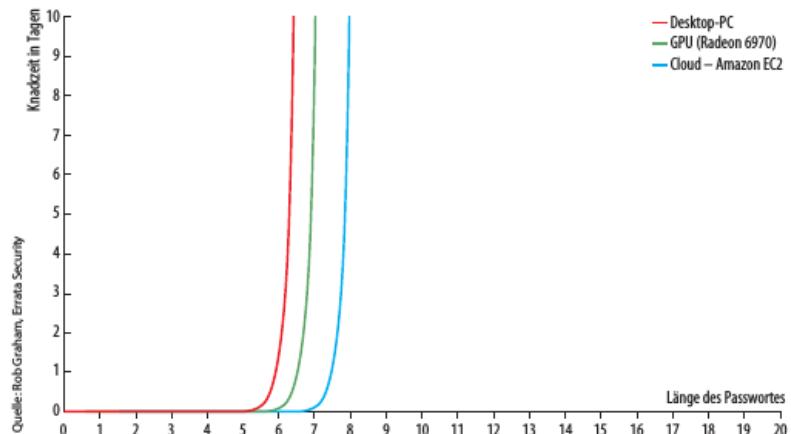
Abbildung 2: Quelle: [Matt Crypto](#)

Eine SHA-1-Operation besteht aus 80 Operationen dieses Typs, gruppiert in 4 Durchläufen mit jeweils 20 Operationen. A,B,C,D,E sind 32-Bit-Register, die zu Beginn mit Konstanten geladen werden.  $F$  ist eine nichtlineare Funktion, die im jeweiligen Durchlauf eingesetzt wird.  $W_t$  bezeichnet einen 32-Bit-Block des Eingabestroms und  $K_t$  eine für jede Operation unterschiedliche 32-Bit-Konstante.  $\lll_s$  bezeichnet die bitweise Linkssrotation um  $s$  Stellen, wobei  $s$  für jede Operation variiert.  $\boxplus$  bezeichnet die Addition

modulo  $2^{32}$ . [Pseudocode](#)

### 3 Passwort-Cracking

Abbildung 3: Knackzeit durch Brute Force. Quelle: c't 3/2013



Gegenmaßnahme: scrypt, iteriertes Hashen

#### Knackzeit durch Brute Force

Der Passwort-Knacker Hashcat schafft auf einer High-End-Grafikkarte (Nvida GeForce GTX 980 Ti) pro Sekunde

- 14 Milliarden MD5-Hashes,
- 4,6 Milliarden SHA-1-Hashes,
- 12000 bcrypt-Hashes



Abbildung 4: Regal mit Grafikkarten

## 25 häufigsten Passwörter und PINs

Nr.	Passwörter	MD5	PINs
1	password	286755fad04869ca523320acce0dc6a4	1234
2	123456	f447b20a7fcfb53a5d5be013ea0b15af	0000
3	12345678	23cd018507b52418db7740ccb5543e54	2580
4	1234	e7df7cd2ca07f4f1ab415d457e6elc13	1111
5	qwerty	a68650deb2742ec3cb41518e26aa2d89	5555
6	12345	d577273f885c3f84dadb8578bb41399	5683
7	dragon	c0ce0dff999ea7d40c1e9a944dd0fc5	0852
8	pussy	7084a436b380898b7ae9a61b8beb6	2222
9	baseball	5b9d07ad9c1bed09d6986e593f4ca7dc	1212
10	football	a174fded30655297e43208a716875b3	1998
11	letmein	4aacf9c858c82716ab0034320bd2fe9	6969
12	monkey	2f548f61bd37f628077e552ae1537be2	1379
13	696969	551cf8e41cdce05871e41a4a8c6b6228	1997
14	abc123	2e69a8abba8b998a1939450e4089ed	2468
15	mustang	d3c3ce1b8b4e88e23a04a1f123eeb593	9999
16	michael	3a8c08f89cfe9a0a564fe3fb277263a	7777
17	shadow	54badf757ad046d8e4e762aae1e022a7	1996
18	master	c963080767f45828c31f83ca5cd25d3e	2011
19	jennifer	9bf4c002560f3f814ed5cdbeba250808	3333
20	111111	77a319564621b96fa0656e24c67960ef	1999
21	2000	f95f8943f6dcf7fb3c1c8c2cab5455f8b	8888
22	jordan	365091b269ec29df8a967a1aefaf648c	1995
23	superman	0cb290b2a9b7d2854e50f6844ac999e4	2525
24	hadley	1f560e3339141244d68e58b47b55ef0	1590
25	1234567	1b504d3328e16fdf281d1fb0516dd90b	1235

Quelle: LinkedIn-Hack, 6,5 Millionen Kennwörter abgegriffen (2012)

Abbildung 5: Häufige Passwörter und PINs

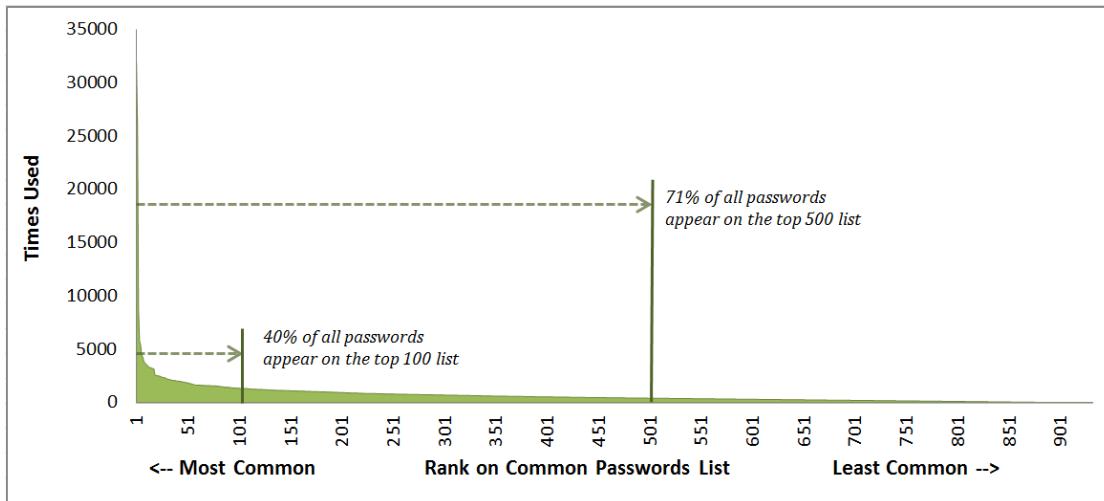


Abbildung 6: 10,000 Top Passwords

## Hash-Tabellen

Ist die Hash-Methode und die maximale Passwortlänge bekannt, könnte man einmalig riesige Tabellen berechnen, um eine Einwegfunktion schnell zu invertieren.

Das Utah Data Center bei Camp Williams der NSA soll bis zu einem Yottabyte =  $10^{24} \approx 2^{80}$  Byte fassen ([Der Spiegel](#)).

### Die Regenbogentabelle / Rainbow Table

ist eine Datenstruktur/ein Algorithmus, um das Urbild einer Einwegfunktion schneller als durch vollständiges Suchen und gleichzeitig mit weniger Speicher als durch vollständige Speicherung zu finden.

### Die Regenbogentabelle / Rainbow Table

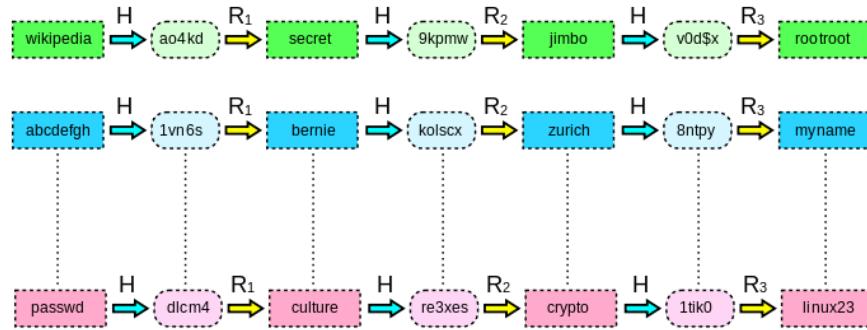


Abbildung 7: Quelle: [Wikipedia](#)

Hierbei ist  $H$  die Hashfunktion und  $R_i$  sind Reduktionsfunktionen, die einen Hashwert wieder auf ein Passwort abbilden.

In diesem Beispiel soll das Passwort zum Hashwert „re3xes“ ermittelt werden.

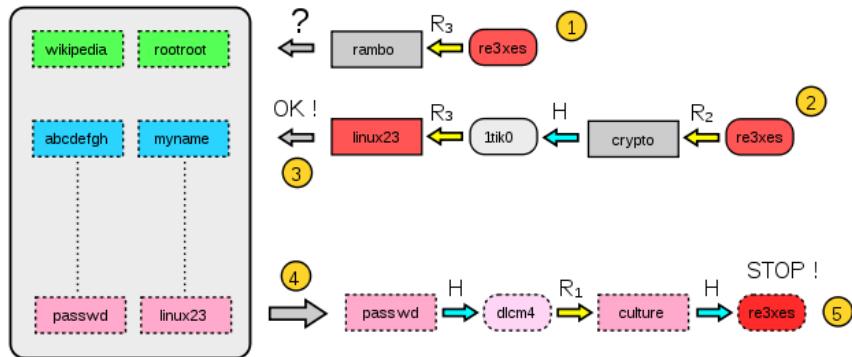


Abbildung 8: Quelle: [Wikipedia](#)

Wie verhindert man den Angriff durch Regenbogentabellen?

### Gegenmaßnahmen

- lange Kennwörter
- Salt: Vor dem Hashen eines Passwortes, wird eine zufällige Zeichenkette (Salt) angehängt. Man speichert nun den Hash und den Salt gemeinsam.

```
root@kali:~# tail /etc/shadow
postgres:*:15772:0:99999:7:::
sshd:*:15772:0:99999:7:::
rtkit:*:15772:0:99999:7:::
snmp:*:15772:0:99999:7:::
stunnel4:*:15772:0:99999:7:::
statd:*:15772:0:99999:7:::
sslh:*:15772:0:99999:7:::
saned:*:15772:0:99999:7:::
Debian-gdm:*:15772:0:99999:7:::
jose:$6$Cqi0wyE$Rutn7Vt7yuALGpkYfFT3p5zqywaMsbk74/u7vz/aIj1Mz3LftQsgUnpFBfVjDv/IMKPBuuIRBd85QrRKv0U1R/:15871:0:99999:7:::
root@kali:~#
```

\$6 = SHA-512 mit vielen Durchläufen, Cq...yE ist Salt

### Gute Passwörter

Ein gutes Passwort besteht aus zufälligen Zeichen.

Grundproblem: Menschen brauchen Muster, um sich Passwörter zu merken — Cracker kennen diese Muster besser als Menschen.

Brute Force Angriffe: Aktuelle Hardware schafft ca. 350 Mrd Windowspasswörter pro Sekunde. D.h. ein 8 Zeichen Passwort würde in etwa 5,5 Stunden geknackt werden.

### Regeln für gute Passwörter

- Für jeden Dienst / jedes Konto ein eigenes Passwort
- Mindestens 8 zufällige Zeichen ⇒ Passwortgeneratoren
- Man lüge auf Fragen zum Passwortreset.
- Man wähle ein besonders gutes Passwort ( $\geq 10$  Zeichen) für das Standard-Email-Konto.
- (Passwörter regelmäßig ändern. Siehe z.B. [BSI für Bürger](#).)

**Beispiel des BSI zur Passwortgenerierung:** „Morgens stehe ich auf und putze mir meine Zähne drei Minuten lang.“ Nur die ersten Buchstaben: MsiaupmmZdMl. i und l sieht aus wie 1, & ersetzt das und: Ms1a&pmmZ3M1.

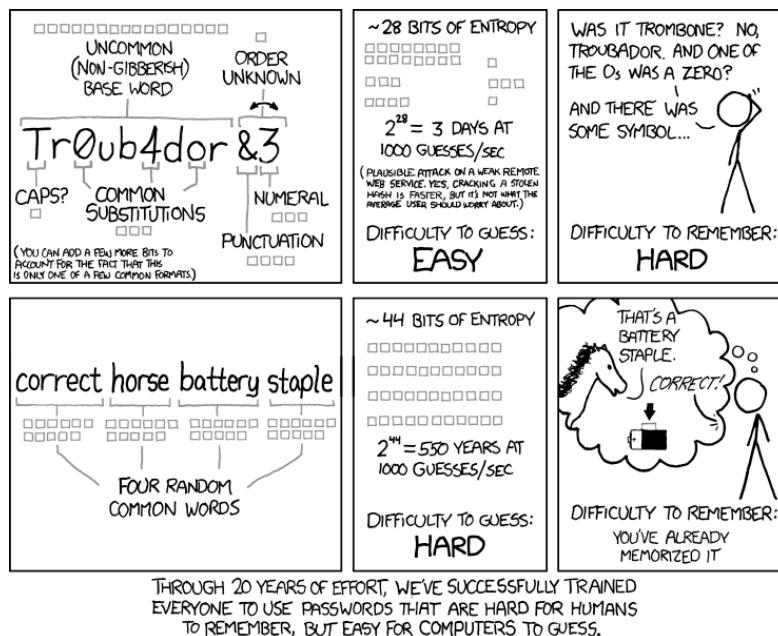


Abbildung 9: Quelle: [xkcd](https://xkcd.com/937/)

#### Apple-ID und Passwort

Geben Sie als Ihre Apple-ID Ihre primäre E-Mail-Adresse ein. Diese E-Mail-Adresse wird als Kontakt-E-Mail-Adresse für Ihren Account verwendet.

Apple-ID

Passwort

Passwort bestätigen

**Sicherheitsfragen**  
Wählen Sie unten drei Sicherheitsfragen aus, um Ihre Identität zu überprüfen, falls Sie Ihr Konto vergessen.

Sicherheitsfrage	Antwort
Sicherheitsfrage	Antwort

Sicherheitsfrage	Antwort
Sicherheitsfrage	Antwort

Sicherheitsfrage	Antwort
Sicherheitsfrage	Antwort

Bitte füllen Sie dieses Feld aus.

**Das Passwort:**

- Muss mindestens einen Kleinbuchstaben enthalten
- Muss mindestens einen Großbuchstaben enthalten
- Muss mindestens eine Ziffer enthalten
- Darf nicht mehrere identische, aufeinander folgende Zeichen enthalten
- Darf nicht mit dem Accountnamen identisch sein
- Muss aus mindestens 8 Zeichen bestehen
- Darf kein häufig verwendetes Kennwort sein

Abbildung 10: Quelle: [Apple](https://www.apple.com/de/account/privacy/)

## Passwortmanager aus Papier



Abbildung 11: Quelle: c't 18, 2014

## Prominente Datendiebstähle

- rambler.ru, 2012, 98 Millionen Passwörter im Klartext
- VK.com, 2012, 171 Millionen Passwörter im Klartext
- Last.fm, 2012, 43 Millionen Passwörter, MD5-Hashes ohne Salz
- LinkedIn.com, 2012, 177 Millionen Passwörter, SHA1 ohne Salz
- Yahoo.com, 2014, 500 Millionen Passwörter, bcrypt
- Badoo.com, 2016, 127 Millionen Passwörter, MD5-Hashes ohne Salz

## Wurden Ihre Anmeldedaten geklaut?

- [';-have i been pwned?](#)
- [Hasso-Plattner-Institut](#)

## Prominente Datendiebstähle

Wenn eine geklauten User-Datenbank öffentlich bekannt wird, liegt der Diebstahl oft Monate oder sogar Jahre zurück. In der Zwischenzeit haben Kriminelle mit den Daten „gearbeitet“.

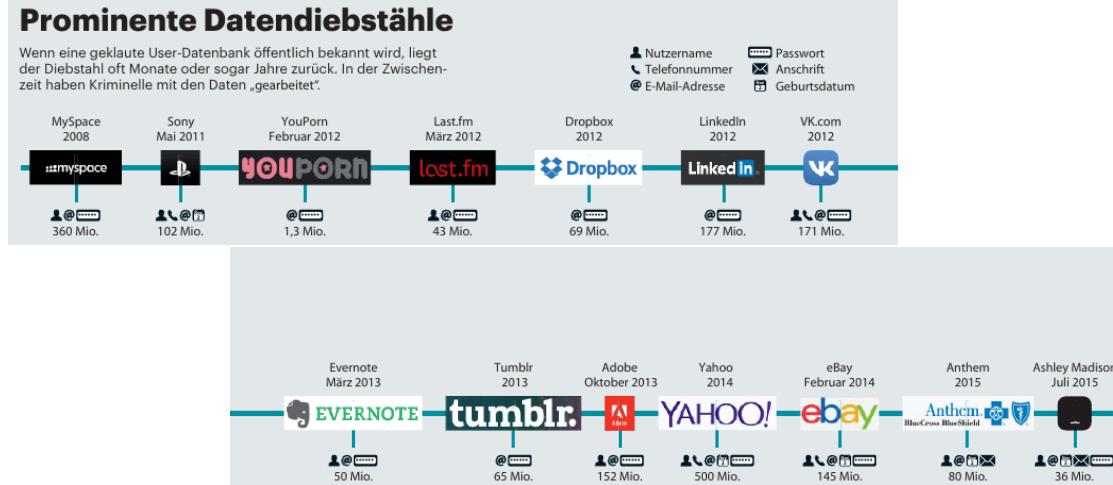
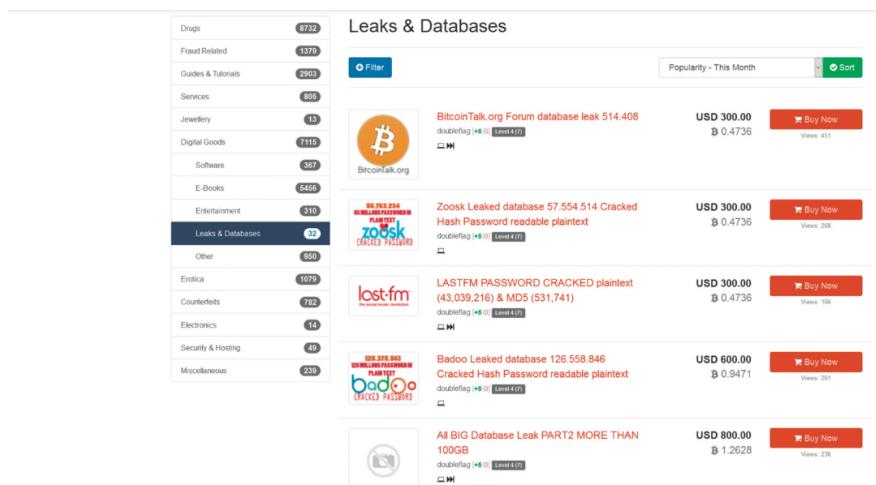


Abbildung 12: Quelle: c't 23/2016

## Marktplatz für geklauten Daten



Im Darknet werden riesige Mengen an geklauten Daten gehandelt – 43 Millionen Last.fm-Passwörter kosten beispielsweise gerade 300 US-Dollar.

Abbildung 13: Quelle: c't 23/2016

## Eine kurze Geschichte des Passwort-Phishings

Banken verschicken Emails mit URL-Links, und so taten dies auch Passwort-Phisher.

- Die erste Empfehlung der Banken war, auf gute Sprache zu achten.
- Die Passwort-Phisher verbesserten die Sprache oder nutzten die Emails der Banken mit ausgetauschten Links.
- Achten Sie auf das Schloss-Symbol des Web-Browsers.

- Also nutzten Phishing-Seiten SSL oder betteten einfach eine Schloss-Grafik ein.
- Einige Banken schrieben die letzten 4 Stellen der Kontonummern in die Email.
- Die Passwort-Phisher verwendeten einfach die ersten 4 Ziffern einer Kontonummer, die oft bei allen Kunden identisch ist.
- Die Banken empfahlen die Maus über die Adresszeile des Browsers zu legen, um die URL zu überprüfen.
- Die Passwort-Phisher verwendeten nicht druckbare Zeichen in der URL.
- ...

### **Andere und ergänzende Authentifizierungen**

- Passwörter auf USB-Token oder Chipkarten + PIN
- 2-Faktor-Authentifizierung: meistens Besitz & Wissen
- biometrische Merkmale / Körpermerkmale (Handgeometrie, Fingerabdruck, Iris, Gesicht, ...)
- Muster und Gesten
- Transponder und Schlüsselkarten

## 4 Hash-Pointer und Datenstrukturen

### Hash-Pointer

Ein *Hash-Pointer* ist ein Zeiger auf einen Datensatz, der zudem noch den Hash-Wert dieser Daten enthält.

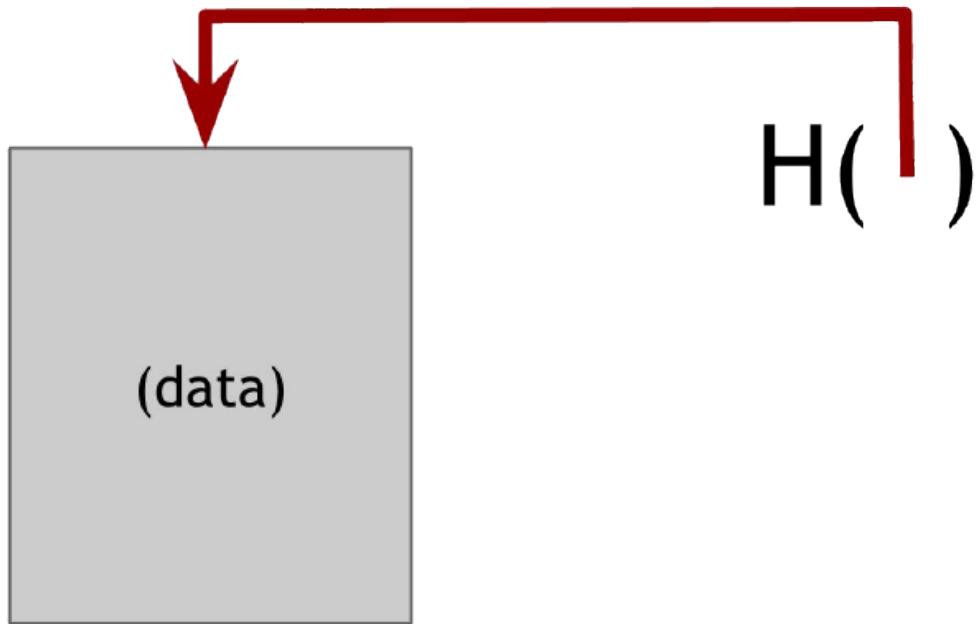


Abbildung 14: Quelle: [Bitcoin and Cryptocurrency Technologies](#)

### Die Blockchain

Eine *Blockchain* ist eine lineare Verkettung von Datensätzen, wobei jeder Datensatz den Hash-Pointer zu dem vorangehenden Datensatz enthält. Der Kopf enthält lediglich einen normalen Hash-Pointer auf den jüngsten Datensatz.

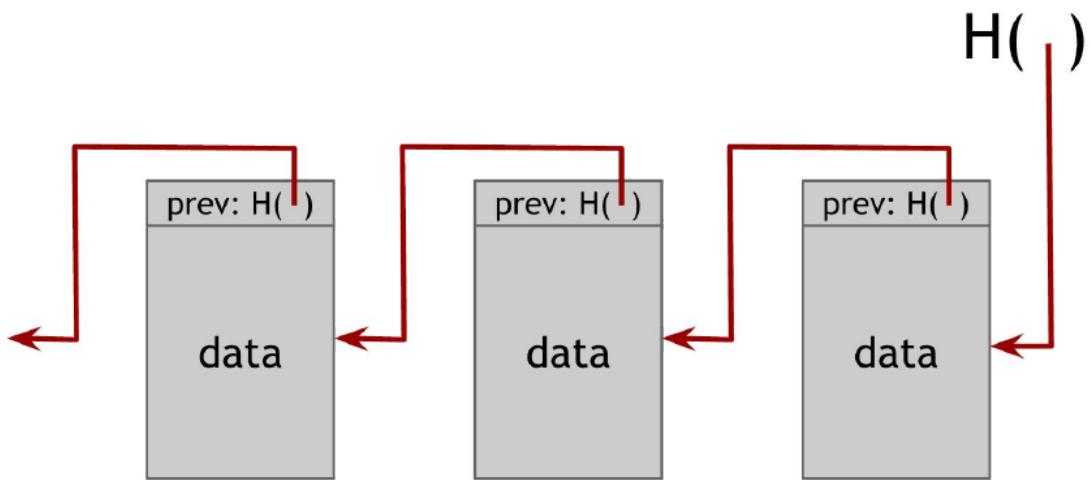


Abbildung 15: Quelle: Bitcoin and Cryptocurrency Technologies

### Anwendung: Manipulationsgesicherte Aufzeichnungen

Die Idee ist, Logdateien (z.B. auf stündlicher Basis) als Blockchain zu speichern, so dass jede Änderung in den Logdateien erkannt werden muss.

Dazu überlegen wir, was passiert, wenn ein Angreifer in der Mitte der Blockchain Daten ändert:

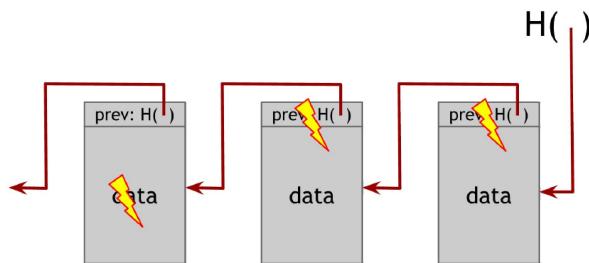


Abbildung 16: Quelle: Bitcoin and Cryptocurrency Technologies

- Die darauf nächste Logdatei (der nächste Block) in der Blockchain enthält einen Hash-Pointer auf diese geänderte Logdatei, d.h. einen Zeiger und den Hashwert, der jetzt nicht mehr stimmt.
- Der Angreifer muss also in dem nächsten Block den Hash-Wert ändern.
- usw.

Um eine Manipulation zu erkennen, muss lediglich der letzte Hash-Pointer besonders vor Änderungen abgesichert sein.

## Bitcoin – Blockchain

Die Kryptowährung Bitcoin speichert die Transaktionsdaten als verteilte Datenbank in einer Blockchain.

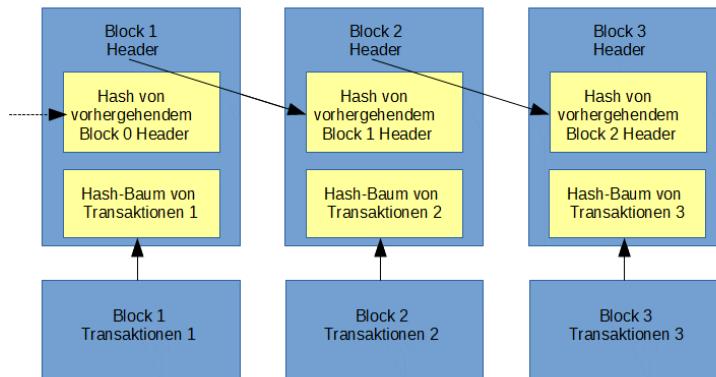


Abbildung 17: Vereinfachte Bitcoin Blockchain

## Bitcoin – Der Arbeitsbeweis

Um in dem Bitcoin-Netzwerk Einigkeit darüber zu erlangen, welcher nächste Block zur Blockchain hinzugefügt wird, muss ein *Arbeitsbeweis* (Proof of Work) erbracht werden. Er ist schwer zu erbringen aber leicht zu überprüfen.

Der Arbeitsbeweis besteht darin folgende Ungleichung zu erfüllen

$$H(\text{Zufall} \parallel \text{Hash-Wert des vorherigen Blocks} \parallel \text{tx} \parallel \dots \parallel \text{tx}) < S$$

Hierbei ist  $H$  eine Hash-Funktion, tx stellen Transaktionen dar, und  $S$  ist ein Zielwert, der so an die Rechenleistung des Bitcoin-Netzwerks angepasst wird, dass im Mittel alle 10 Minuten ein neuer Block zur Blockchain hinzugefügt wird. Man spricht auch vom Bitcoin-schürfen, weil diese Arbeit entlohnt wird.

**Gibt es mehrere unterschiedliche korrekte Blockchains, dann gilt immer die längere.**

## Bitcoins

- Bitcoins können lokal auf Computern oder Smartphones gespeichert werden.
- Bitcoins können von einem Dienstleister verwaltet werden.
- Jeder kann beliebig viele Konten eröffnen.
- Bitcoins können zu jeder Zeit digital verschickt werden.
- Der kleinste Betrag in Bitcoin ist 0,00000001 (= 1 Satoshi)
- Bitcoins werden von keinem Staat reguliert, es gibt keine Zentralbank.

- Bitcoins werden dezentral verwaltet.

## Geschichte

- Bitcoin wurde von Satoshi Nakamoto 2008 erfunden.
- Die erste Transaktion fand am 01.03.2009 statt.
- Angesteckt vom großen Erfolg von Bitcoin, sind 2011 weitere vom Bitcoin-Quellcode abgeleitete Währungen gestartet.
- Ab 2013 bieten immer mehr populäre Webseiten Bitcoin als Zahlungsmöglichkeit an.

Abbildung 18: Anzahl der Bitcoin-Transaktionen pro Tag



## Ablauf einer Bitcoin-Transaktion / Überweisung

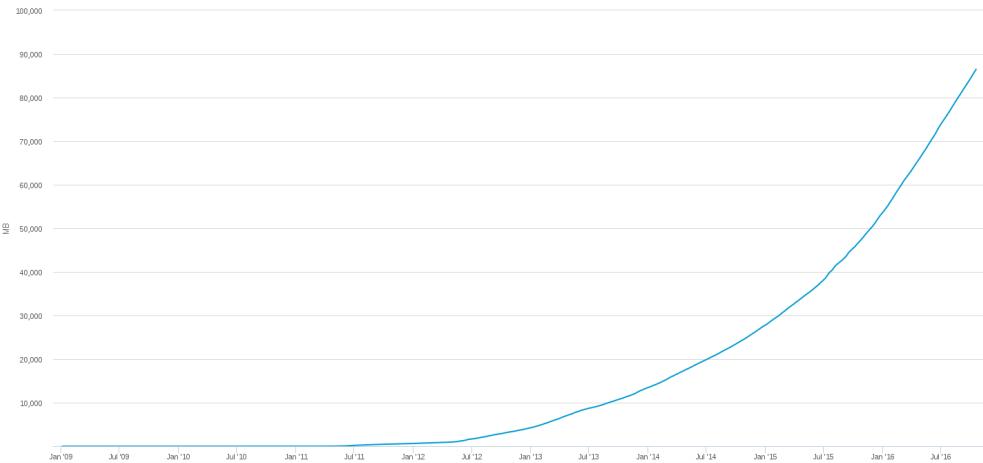
1. Man benötigt die Adresse des Empfängers (Kontonummer).
2. Man füllt die Überweisung aus (VON → NACH).
3. Man sendet die Überweisung in das Bitcoin-Netzwerk.
4. Die Überweisung wird auf Gültigkeit geprüft.
5. Die Überweisung wird im Bitcoin-Netzwerk an alle Teilnehmer verteilt.
6. Die Überweisung wird in das Transaktionsverzeichnis eingebaut.
7. Die Kontostände sind damit aktualisiert.

## Die Blockchain / Das Transaktionsverzeichnis

- In der Blockchain sind alle jemals durchgeföhrten Transaktionen verzeichnet.
- Die Blockchain ist auf allen Rechnern des Bitcoin-Netzwerkes gespeichert.

- Neue Transaktionen werden in Blöcken gesammelt, und ein fertiger Block der Blockchain hinzugefügt.

Abbildung 19: Wachstum der Blockchain über die Jahre



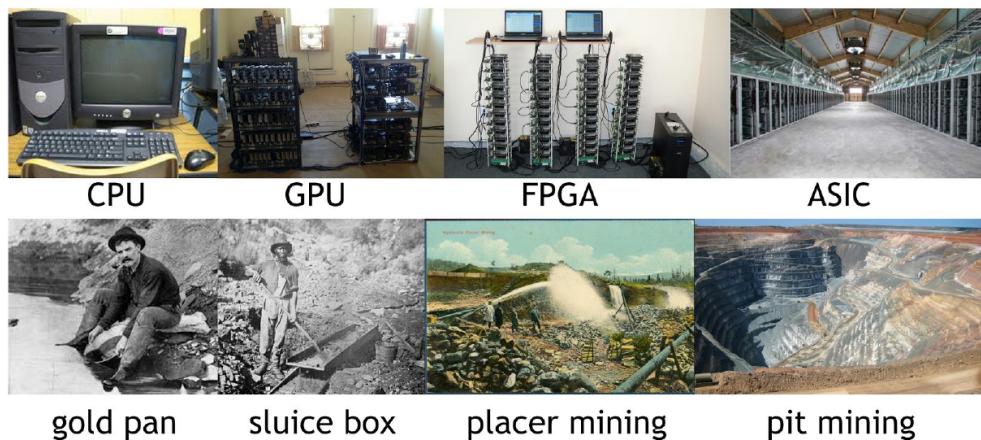
### Woher kommen Bitcoins?

- Für jeden fertigen Block, der zur Blockchain hinzugefügt wurde, erhält der Ersteller eine Belohnung.
- Die Fertigstellung des ersten Blocks wurde mit 50 Bitcoins belohnt. Alle 210000 Blöcke halbiert sich Belohnung.

$$\text{Bitcoin Obergrenze} = 210000 \cdot (50 + 25 + 12.5 + \dots) \approx 21 \text{ Millionen}$$

- Man kann keine Schulden in Bitcoins machen.

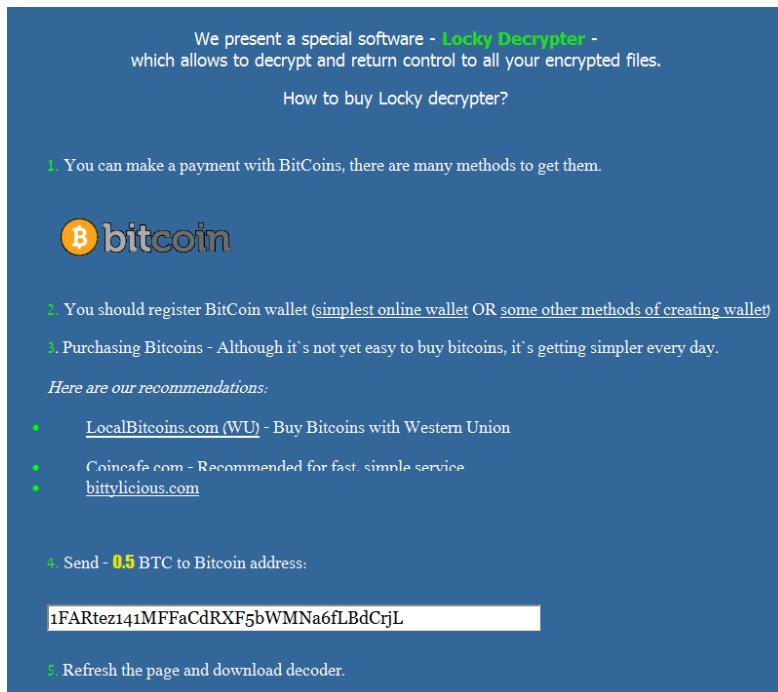
Abbildung 20: Evolution des Schürfens



### Wie kommt man an Bitcoins?

- Bitcoin-Schürfer müssen Stromrechnungen bezahlen und wollen die geschürften Bitcoins gegen andere Währungen tauschen.
- Es gibt Börsen zum Kaufen und Verkaufen von Bitcoins.
- Es gibt Handelsplätze die Käufer und Verkäufer persönlich zueinander bringen.

Abbildung 21: Bitcoins und Internetkriminalität



## Bitcoins und Anonymität

- Alle Transaktionen sind öffentlich.
- Man kann für jede Transaktion ein neues Konto eröffnen.
- Beim Verkaufen oder Kaufen an Börsen muss man sich ausweisen.
- Es gibt Bitcoin-Mixer zur Erhöhung der Anonymität.

## 5 Nachrichtenauthentifizierung

### Nachrichtenauthentifizierung

Ein Nachrichtenauthentifizierungscode (Message-Authentiation-Codes, kurz MAC) berechnet eine kryptographisch sichere Prüfsumme für eine gegebene Nachricht.

Die Prüfsumme hängt dabei von einem geheimen Schlüssel  $K$  ab.

#### Eigenschaften

- MACs akzeptieren Nachrichten beliebiger Länge.
- Die Länge der Prüfsumme ist konstant.
- Es wird die Integrität und Authentizität einer Nachricht sichergestellt.
- Die Authentizität des Absenders kann nicht bewiesen werden.

### Message Authentication Codes (MACs)

Zusammen mit dem Klartext  $M$  wird also auch immer der MAC mitgeschickt  $\text{MAC}_K(M)$ .

Der Empfänger überprüft, ob  $\text{MAC}_K(M)$  stimmt.

Informationstheoretisch erfüllt ein sicherer MAC

$$P(M \mid \text{MAC}_K(M)) = P(M)$$

In der Praxis soll ein Angreifer einen korrekten MAC also nur raten können. Hat ein MAC eine Länge von  $n$  Bits, so liegt seine Erfolgswahrscheinlichkeit bei  $2^{-n}$ .

**Vorsicht:** MACs schützen nicht vor Replay-Angriffen!

### Keyed-Hash Message Authentication Code (HMAC)

Ein Keyed-Hash Message Authentication Code (HMAC) ist ein Nachrichtenauthentifizierungscode, dessen Konstruktion auf einer Hash-Funktion basiert.

Wir stellen den HMAC nach [RFC 2104](#) vor.

Gegeben sei eine Nachricht  $M$  und ein geheimer Schlüssel  $K$  und eine Hash-Funktion  $H$  mit der Blocklänge  $b$  (z.B. ist  $b = 64$  Byte für den SHA-1).

Dann ist der HMAC definiert als

$$\text{HMAC}_K(M) = H\left((K \oplus \underbrace{0x5C \dots 0x5C}_{b\text{-mal}}) \parallel H\left((K \oplus \underbrace{0x36 \dots 0x36}_{b\text{-mal}}) \parallel M\right)\right)$$

Hierbei ist der  $\parallel$ -Operator die Konkatenation von Zeichenketten und  $\oplus$  die XOR-Operation.  $K$  wird durch das Auffüllen mit Nullbytes auf die Blocklänge  $b$  gebracht.

## Keyed-Hash Message Authentication Code (HMAC)

Das folgende Beispiel wurde in [Python](#) implementiert.

```
>>> import hmac
>>> import hashlib
>>> h = hmac.new(key=b'geheim', msg=b'nachricht', digestmod=hashlib.sha1)
>>> h.hexdigest()
'407478ea477f55c229cba6a0026316d41adb1d83'
```

Zum Testen siehe auch [HMAC Generator / Tester Tool](#).

**Bemerkung:** Die Sicherheit der Konstruktion des HMAC setzt keine Kollisionsresistenz der Hash-Funktion  $H$  voraus, siehe [M. Bellare, New Proofs for NMAC and HMAC: Security without Collision-Resistance](#).

Die Konstruktion des HMAC  $\text{HMAC}_K(M) = H\left((K \oplus C_1) \parallel H((K \oplus C_2) \parallel M)\right)$  mutet umständlich an. Der Grund ist, dass fast alle Hash-Funktionen auf der Merkle-Damgård-Konstruktion beruhen:

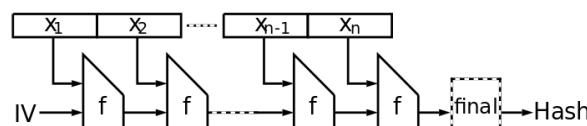


Abbildung 22: Merkle-Damgård-Konstruktion

Angenommen der MAC wäre  $H(K \parallel M)$ . Ist die Finalisierungsfunktion leicht umkehrbar, so kann zu einer beliebigen Nachricht  $X$  leicht  $H(K \parallel M \parallel X)$  berechnet werden.

Vertauscht man die Reihenfolge des Schlüssels  $K$  und der Nachricht  $M$ , d.h. man konstruiert den MAC als  $H(M \parallel K)$ , so scheint das Problem einer schwachen Finalisierungsfunktion nicht zu bestehen.

Ist allerdings  $H$  nicht kollisionsresistent, so wäre es einem Angreifer eventuell möglich zu einer Nachricht  $M$  eine zweite Nachricht  $M_0$  mit demselben Hashwert zu berechnen, d.h.  $H(M) = H(M_0)$ .

Wegen der iterativen Struktur des Hashfunktion ist damit auch

$$H(M \parallel K) = H(M_0 \parallel K)$$

## 6 Wechselcodeverfahren

### Einmalkennwörter

Jedes Einmalkennwort ist nur für eine einmalige Verwendung gültig.

- Lauschen ist zwecklos.
- Replay-Attacken sind unmöglich.

### Verfahren

- Kennwortlisten (TAN-Listen)
- Kennwortgeneratoren
  - Zeitgesteuerte Generatoren (TOTP z.B. Google Authenticator, [SecureID-Tokens](#))
  - Ereignisgesteuert (HOTP)
  - Challenge-Response-gesteuert

### Challenge-and-Response-Verfahren

Die Authentifizierung mit einem Passwort weist die grundlegende Schwäche auf, dass nicht nur das Geheimnis statisch ist, sondern auch immer die gleichen Nachrichten gesendet werden.

Als alternative dienen Challenge-Response-Verfahren. Hierbei erzeugt der Server eine Challenge, die vom Client nur gelöst werden kann, wenn er das gemeinsame Geheimnis  $K$  kennt.

Ein einfaches Verfahren mit einer kryptographischen Hash-Funktion  $H$  könnte so aussehen:

1. Der Server sendet eine Zufallszahl, die Challenge,  $C$  an den Client.
2. Der Client berechnet  $R = H(K \parallel C)$  und schickt  $R$  an den Server.
3. Der Server berechnet seinerseits  $R' = H(K \parallel C)$ . Er überprüft ob  $R' = R$ ?

### Ein Wechselcodeverfahren

Bei folgendem Wechselcodeverfahren zur Authentifizierung eines Clients braucht der Server das Geheimnis vom Client nicht zu kennen.

Sei  $H$  eine kryptographische Hash-Funktion,  $h_0$  ein geheimer Startwert und  $n$  die Obergrenze der möglichen Authentifizierungen, z.B.  $n = 1000000$ .

Der Client Berechnet  $h_{i+1} = H(h_i)$  für  $i = 0, \dots, n - 1$ .

Zur Registrierung teilt der Client dem Server  $h_n$  mit.

Will der Client sich gegenüber dem Server authentifizieren, so sendet er  $h_{n-1}$ , und der Server überprüft  $h_n = H(h_{n-1})$ .

Für eine erneute Authentifizierung sendet der Client  $h_{n-2}$ , und der Server überprüft  $h_{n-1} = H(h_{n-2})$ .

usw.

## HMAC-basierte Einmalkennwörter (HOTP)

### HOTP: HMAC-Based One-Time Password Algorithm – [RFC 4226](#)

Sei  $K$  ein Geheimnis,  $C$  ein Zähler

1. Berechne  $S = \text{HMAC}_K(C)$ , wobei als Hash-Funktion SHA-1 verwendet wird.

2. Selektiere 4 Bytes und lösche das höchstwertigste Bit

$$B_{31} = (S[i] \parallel S[i+1] \parallel S[i+2] \parallel S[i+3]) \& 0x7FFFFFFF, \quad i = S[19] \& 0x0F$$

3. Nehme die  $6 \leq d \leq 8$  niederwertigsten Dezimalstellen als Einmalkennwort

$$\text{HOTP}_K(C) = B_{31} \bmod 10^d$$

Im zeitgesteuerten Verfahren TOTP wird der Zähler  $C$  durch einen Zeitzähler ersetzt.

## Yubikey – HOTP

Beispielausgabe:

```
ubnu24252879320780
ubnu24252879088464
ubnu24252879080365
ubnu24252879260232
ubnu24252879900692
ubnu24252879412678
ubnu24252879149293
ubnu24252879448924
```



Abbildung 23: Yubikey

ubnu24252879 = OATH Token Identifier, siehe [Using YubiKey](#).

## 7 Historische Verschlüsselungsverfahren

### Historische Verschlüsselungsverfahren

Die Skytale (vor 2500 Jahren)



Abbildung 24: Quelle: [Wikipedia](#)

Was ist das **Geheimnis**? Durchmesser des Stabs

**Prinzip:** Transposition

**Aufgabe:** Wie lautet die Botschaft der geheimen Nachricht „K R C I O G H N M E B X M N E D M N R K O A“

**Cäsar-Chiffre**



Abbildung 25: Quelle: [WDR](#)

**Geheimnis:** Stellung der Scheiben zueinander, **Prinzip:** Substitution

### Verschlüsselung

#### Allgemeine Permutation eines Alphabets

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

**Geheimnis:** Tabelle (26! Möglichkeiten), **Prinzip:** Substitution

**Angriff:** statistische Analyse (Quelle: A. Beutelspacher, Kryptologie, Vieweg)

Buchstabe	E	N	I	S	R	A	T	D
Häufigkeit (%)	17.4	9.8	7.6	7.6	7.0	6.5	6.2	5.1

#### Automatischer Angriff auf Cäsar-Chiffre

Sei  $p_i$  die relative Häufigkeit des  $i$ -ten Buchstabens im deutschen Alphabet, dann ist

$$\sum_{i=0}^{25} p_i^2 \approx 0.077$$

Sei weiter  $q_i$  die relative Häufigkeit des  $i$ -ten Buchstabens einer Cäsar-Chiffre eines deutschen Klartextes, dann bildet man

$$I_j = \sum_{i=0}^{25} p_i \cdot q_{i+j \bmod 26}$$

für  $j = 0, \dots, 25$ . Gesucht ist dann das  $j$ , für dass  $I_j \approx 0.077$  ist. Dieses  $j$  ist dann der Schlüssel für die Cäsar-Chiffre.

## Historische Verschlüsselungsverfahren

### Vigenère-Chiffre

Klartext-	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
alphabet																										
0	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
1	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
2	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
3	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
4	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
5	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
6	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
7	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
8	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
9	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
10	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
11	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
12	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
13	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
14	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
15	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
16	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
17	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
18	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
19	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
20	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
21	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
22	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
23	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
24	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
25	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 26: Vigenère-Quadrat

**Geheimnis:** gemeinsames Codewort

**Prinzip:** Substitution

**Angriff:** Wie bei Cäsar, wenn die Länge des Codewortes bekannt ist.

Verbesserung durch **Autokey-Verschlüsselung**: Dem Schlüssel wird der Klartext angehängt.

**Aufgabe:** Der Klartext lautet VOMEISEBEFREIT, das Schlüsselwort CAESAR. Wie lautet die Verschlüsselung? XOQWIKGBIXRVKT

### Kryptanalyse der Vigenère-Chiffre (1/2)

Sei  $c_1c_2\dots$  eine Vigenére-Chiffre mit einem Schlüssel der Länge  $k$ .

Um  $k$  herauszufinden, wird, beginnend mit  $l = 1$ , nur jeder  $l$ -te Buchstabe der Chiffre berücksichtigt. Für die Chiffre  $c_1c_{1+l}c_{1+2l}\dots$  wird die Buchstabenhäufigkeit  $q_0, \dots, q_{25}$  bestimmt. Dann berechnet man

$$S_l = \sum_{i=0}^{25} q_i^2$$

Nun können 2 Fälle auftreten:

### Kryptoanalyse der Vigenére-Chiffre (2/2)

1.  $S_l \approx 0.077$ , d.h. es ist  $k = l$  oder der erste Buchstabe des Schlüssels ist gleich dem  $l$ -ten Buchstaben.
2.  $S_l \approx \sum_{i=0}^{25} \left(\frac{1}{26}\right)^2 \approx 0.038$ , d.h. der Text sieht zufällig aus.  $l$  wird um 1 erhöht.

Man kann die Vermutung für  $k = l$  weiter verifizieren, wenn man die Chiffren  $c_2c_{2+l}c_{2+2l}\dots, c_3c_{3+l}c_{3+2l}\dots$  usw. untersucht.

### Kryptoanalyse der Vigenére-Chiffre

Nach Friedman ist die Länge  $k$  des Codewortes eines deutschen Textes der Länge  $n$

$$k \approx \frac{0.0377n}{(n-1) \cdot I - 0.0385 \cdot n + 0.0762}$$

Hierbei ist  $I$  der Friedmansche Koinzidenzindex

$$I = \frac{1}{n(n-1)} \sum_{i=1}^{26} n_i(n_i - 1),$$

wobei  $n_i$  die Häufigkeit des  $i$ -ten Buchstabens des Geheimtextes ist.

### Herleitung der Kryptoanalyse nach Friedman (1/3)

Die Anzahl aller Buchstabenpaare ist  $n(n-1)/2$ .

Die Anzahl aller Paare des Buchstabens a ist  $n_1(n_1 - 1)/2$ , usw.

Die Anzahl der Paare aus gleichen Buchstaben ist  $\sum_{i=1}^{26} n_i(n_i - 1)/2$ .

Die Wahrscheinlichkeit, dass ein beliebiges Buchstabenpaar aus gleichen Buchstaben besteht ist also

$$I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)/2}{n(n-1)/2} = \frac{1}{n(n-1)} \sum_{i=1}^{26} n_i(n_i - 1)$$

### Herleitung der Kryptoanalyse nach Friedman (2/3)

Für lange deutsche Texte gilt  $I \approx I_d = 0.0762$ . (Für englische 0.068.)

Für zufällige Texte ist

$$I = I_r = \frac{1}{n(n-1)} \sum_{i=1}^{26} \frac{n}{26} \left( \frac{n}{26} - 1 \right) \approx \frac{1}{n^2} \cdot \frac{n^2}{26} = \frac{1}{26} = 0.0385$$

Angenommen, die Schlüssellänge ist  $k$ , dann gibt es  $k$  Teiltexte mit etwa  $n/k$  Buchstaben, also jeweils

$$\frac{1}{2} \cdot \frac{n}{k} \cdot \left( \frac{n}{k} - 1 \right) \cdot k = \frac{n(n-k)}{2k}$$

Paare aus diesen Teiltexten und ...

### Herleitung der Kryptoanalyse nach Friedman (3/3)

...und

$$\frac{n(n-1)}{2} - \frac{n(n-k)}{2k} = \frac{n^2(k-1)}{2k}$$

Paare aus verschiedenen Teiltexten.

Die Anzahl der Paare aus gleichen Buchstaben ist also

$$\frac{n(n-k)}{2k} \cdot I_d + \frac{n^2(k-1)}{2k} \cdot I_r$$

Dividiert durch die Anzahl der Paare insgesamt liefert die Wahrscheinlichkeit für Paare aus gleichen Buchstaben, was ungefähr dem Koinzidenzindex  $I$  sein muss. Also

$$I \approx \frac{n-k}{(n-1)k} \cdot I_d + \frac{n(k-1)}{(n-1)k} \cdot I_r \quad \Leftrightarrow \quad k \approx \frac{(I_d - I_r)n}{(n-1)I + I_d - nI_r}$$

## Verschlüsselung

### Formale Definition

Eine Verschlüsselungsfunktion ist eine injektive Abbildung, die einen Klartext in einen Geheimtext abbildet:

$$f_k : M \rightarrow C$$

Sie wird parametrisiert durch eine Menge von Schlüsseln  $k \in K$ .

### Kerckhoffs Prinzip

Man geht davon aus, dass ein Angreifer die Verschlüsselungsfunktion kennt. Das heißt, die Sicherheit der Verschlüsselungsfunktion soll nur vom Schlüssel abhängen.

## **Was macht eine gute Verschlüsselung aus?**

- *Es soll für einen Angreifer unmöglich sein den Schlüssel herauszubekommen.* Hört sich sinnvoll an, greift aber zu kurz:  $f_k(m) = m$ , und eigentlich soll ja die Nachricht geschützt werden. Der Schlüssel  $k$  ist nur Mittel zum Zweck.
- *Es soll für einen Angreifer unmöglich sein den Klartext vollständig zu erhalten.* Selbst wenn er 90% nicht aufdecken kann, kann das fatal sein.
- *Es soll für einen Angreifer unmöglich sein auch nur ein Zeichen zu erhalten.* Manchmal gibt es subtilere Muster in der Chiffre, die es z.B. bei einer verschlüsselten Gehaltsliste doch erlauben herauszufinden, ob A oder B mehr verdient.
- *Egal wie viel Informationen ein Angreifer bereits hat, die Chiffre zu einem Klartext soll keine weiteren Informationen preisgeben.*

## **Verschiedene Angriffsarten**

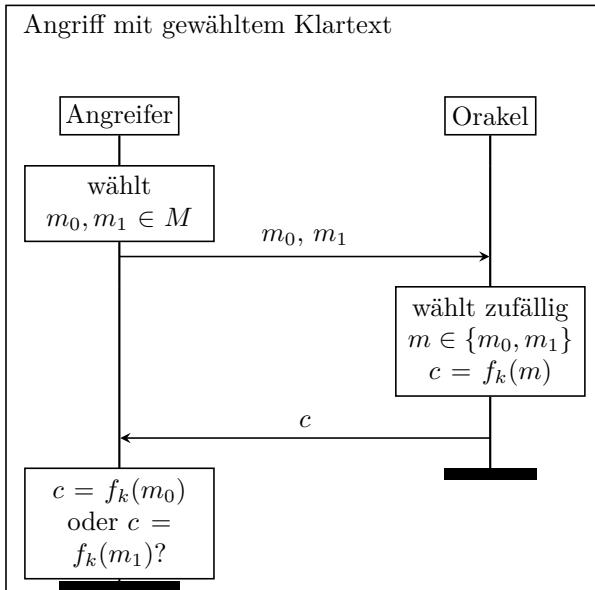
Man unterscheidet folgende Angriffsarten auf eine Verschlüsselung:

- Angriff mit bekanntem Geheimtext
  - Angreifer verfügt über bekannte Geheimtexte.
- Angriff mit bekanntem Klartext
  - Zu den Geheimtexten sind Teile des Klartextes bekannt, z.B. bei einem Brief.
- Angriff mit gewähltem Klartext
  - Angreifer kann sich beliebige Klartexte verschlüsseln lassen.
- Angriff mit gewähltem Geheimtext
  - Angreifer kann sich beliebige Geheimtexte entschlüsseln lassen.

Die Angriffsarten unterscheiden sich in der Menge der Information, die ein Angreifer gewinnt. Die letzten beiden sind adaptiv und setzen ein Orakel voraus.

Umso weniger Informationen eine Chiffre bei einem starken Angriff preisgibt, umso widerstandsfähiger ist sie gegenüber Angriffen.

## **Sicher im Sinne der Ununterscheidbarkeit**



Wenn der Angreifer beim Raten eine Erfolgswahrscheinlichkeit von 0.5 hat, so heißt die Verschlüsselungsfunktion  $f_k$  *sicher im Sinne der Ununterscheidbarkeit*.

### Perfekt sichere Verschlüsselung

Eine Verschlüsselungsfunktion heißt *perfekt sicher*, wenn

$$P(m) = P(m | c) \quad \text{für alle } k \in K, m \in M \text{ und } c = f_k(m).$$

D.h. bei beliebigem Klartext und zugehörigem Chiffretext ist die a-priori-Wahrscheinlichkeit für den Klartext gleich der a-posteriori-Wahrscheinlichkeit für den Klartext, wenn auch der Chiffretext bekannt ist.

### Bemerkung

Eine Verschlüsselungsfunktion ist genau dann *perfekt sicher*, wenn sie auch *sicher im Sinne der Ununterscheidbarkeit* ist.

### Das One-Time-Pad – Die perfekte Verschlüsselung

#### Echte Zufallszahlenfolgen

Sei  $X \in \{0, 1\}$  eine stochastisch unabhängige Zufallsvariable mit

$$P(X = 0) = P(X = 1) = 0.5.$$

Dann erzeugt  $X$  eine echte Zufallsbitfolge.

#### Das One-Time-Pad

Sei  $m \in \{0,1\}^n$  mit  $m = (m_1, \dots, m_n)$  eine Binärnachricht der Länge  $n$  und  $k = (k_1, \dots, k_n)$  eine echte Zufallsbitfolge. Dann erzeugt das One-Time-Pad den Chiffretext  $c = (c_1, \dots, c_n)$  durch

$$c_i = m_i \oplus k_i, \quad i = 1, \dots, n.$$

Das One-Time-Pad bietet perfekte Sicherheit.

### Bemerkungen

- Jeder Schlüssel darf nur ein einziges Mal verwendet werden.  
Aus  $c_1 = k \oplus m_1$  und  $c_2 = k \oplus m_2$  folgt  $c_1 \oplus c_2 = m_1 \oplus m_2$   
Aufgabe: Bekannt ist  $(m_1, c_1) = (1101, 0110)$  und  $c_2 = 1100$ . Wie lautet der Klartext  $m_2$ ?
- Wann ist die Vigenére-Chiffre perfekt sicher?  
Das Kennwort muss zufällig und ebenso lang wie der Klartext sein (siehe Vernam-Chiffre).
- Wann ist die Cäsar-Chiffre perfekt sicher?
- Ist das One-Time-Pad praktikabel?

### Quantenkryptographie

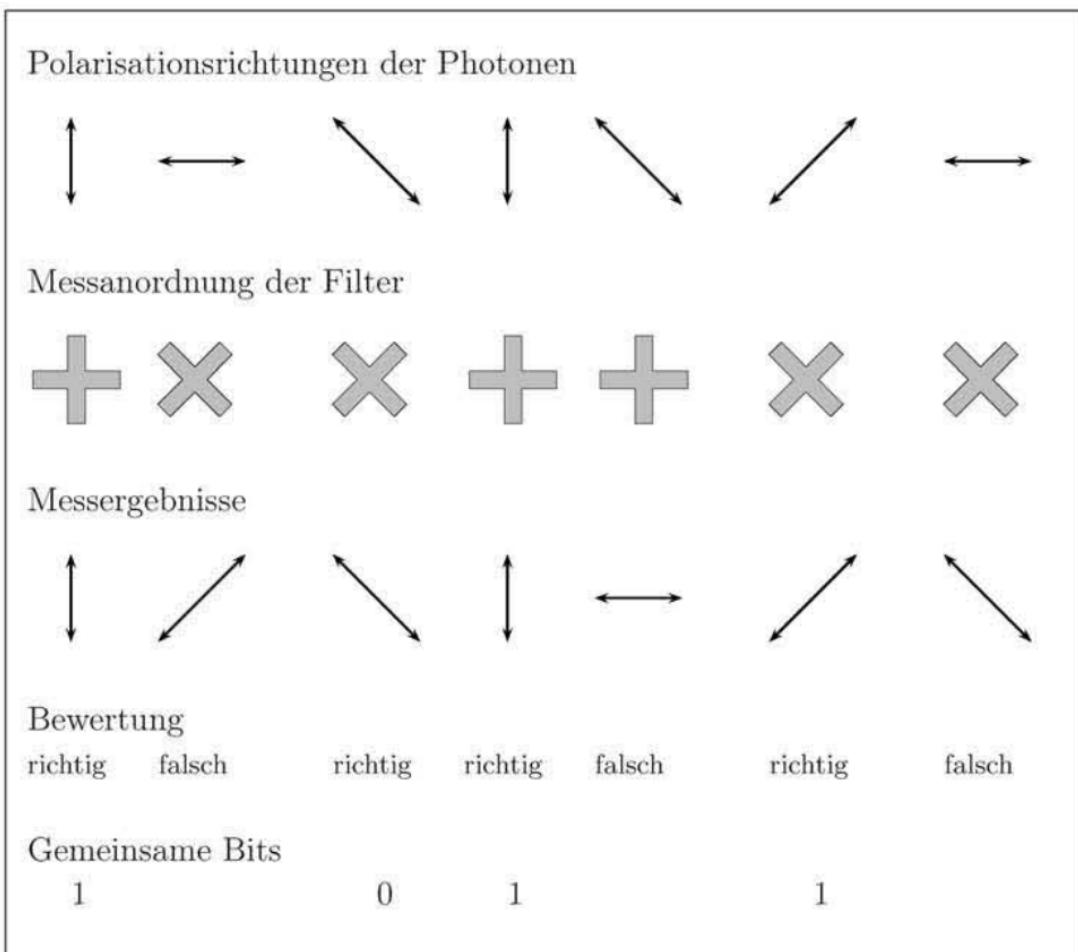


Abbildung 27: Quelle: Beutelspacher et al.: Kryptografie in Theorie und Praxis

- Nach der Übertragung verifizieren Empfänger und Sender, wann die Messanordnung gestimmt hat. Im Mittel bleibt die Hälfte der Bits übrig.
- Von den übrig gebliebenen Bits werden einige Bits aufgedeckt, um einen Lauscher zu enttarnen. Der Lauscher wird mit der Wahrscheinlichkeit 0.25 pro Bit enttarnt.

### Schlüsselaustausch per Satellit

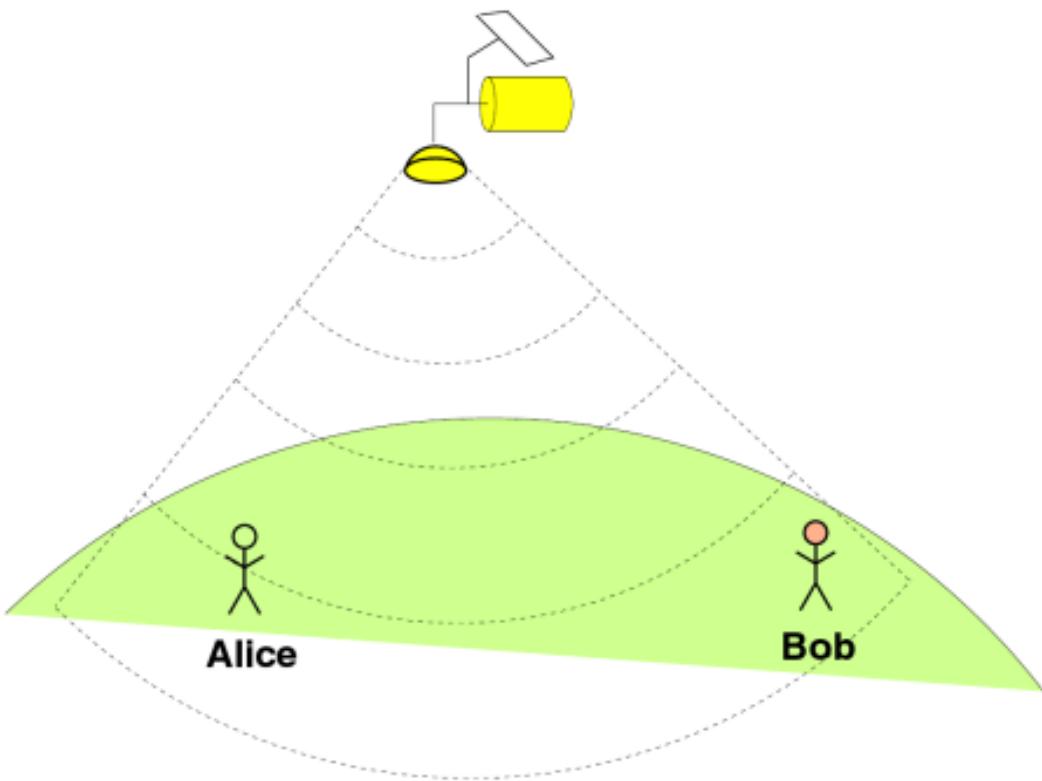


Abbildung 28: Quelle: Ertel: Angewandte Kryptographie, Hanser

- Ein Satellit erzeugt einen zufälligen Bitstrom mit einer extrem hohen Datenrate.
- Die extrem hohe Datenrate verhindert das Speichern des kompletten zufälligen Bitstrom über einen längeren Zeitraum.
- Alice und Bob sprechen sich ab, welche Bits des Datenstroms sie später in einem One-Time-Pad verwenden wollen.

## Verschlüsselung

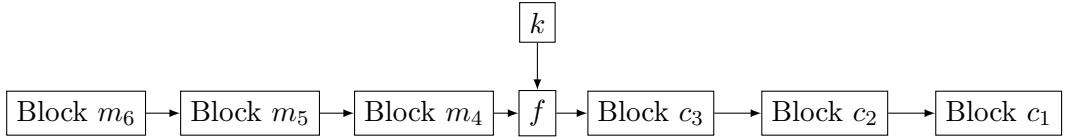
### Praktische Verfahren

- Praktische Verfahren liegen irgendwo zwischen der Cäsar-Chiffre und dem One-Time-Pad.
- Man versucht das One-Time-Pad nachzuahmen:
  - Man versucht einen möglichst langen zufälligen Schlüssel zu generieren → Stromchiffre.
  - Man verwendet einen relativ kurzen Schlüssel immer wieder und investiert in komplexe Verschlüsselungsfunktionen → Blockchiffre.

## 8 Verschlüsselung durch Blockchiffren

### Blockchiffren

Klartexte werden blockweise mit dem Schlüssel verknüpft:



Ist der Klartext zu kurz bzw. kein Vielfaches der Blocklänge, so muss der letzte Block aufgefüllt werden, bis er die passende Länge hat. Man nennt dies *Padding*.

Blockchiffren haben 2 wichtige Parameter:

- Die Blocklänge.
- Die Schlüssellänge.

Im Jahr 1929 erfand der Mathematiker Lester Hill eine Blockchiffre mit variabler Blocklänge für das Alphabet a,b,c,...,z,ä,ö,ü, wobei für die Buchstaben die Zahlen 0 bis 28 benutzt werden. Für die Blocklänge 2 besteht der Schlüssel aus einer  $(2 \times 2)$ -Matrix

$$K = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}.$$

Jeder Klartextblock  $m$  in Form eines Spaltenvektors der Länge 2 wird chiffriert durch

$$c = K \cdot m,$$

wobei alle Additionen und Multiplikationen modulo 29 zu berechnen sind.

1. Es sei ein Klartext-Geheimtext-Paar bekannt. Der Geheimtext zu `turing` sei UÖIHGH. Wie lautet die Schlüsselmatrix  $K$ ?
2. Warum wäre ein Alphabet mit 26 Buchstaben keine gute Wahl?

Klartextblöcke und Geheimtextblöcke haben dieselbe Größe  $n$ , typische Werte sind 64, 128, 256 Bit.

Eine Blockchiffre ist eine Permutation auf der Menge der Zeichenketten der Länge  $n$ . Mit jedem Schlüssel  $k \in K$  wählt man eine „geeignete“ Permutation aus.

Es gibt  $(2^n)!$  verschiedene Permutationen von  $n$ -Bit Blöcken. Ist die Schlüssellänge ebenfalls  $n$ , so gibt es  $2^n$  verschiedene Verschlüsselungsfunktionen/Permutationen.

Schwäche von Blockchiffren: Gleiche Klartextblöcke werden auf gleiche Geheimtextblöcke abgebildet.

Abhilfe: Diverse Betriebsarten von Blockchiffren.

## Allgemeine Angriffe auf Blockchiffren

### Vollständige Schlüsselsuche

Es wird ein Klartext-Geheimtext-Paar  $(m, c)$  benötigt. Dann wird für alle Schlüssel  $k \in K$  getestet, ob  $f_k(m) = c$ . Am Ende gibt es mindestens einen Kandidaten.

Charakteristik: viel Rechenaufwand, geringer Speicherbedarf

### Wertetabellen

- Der Angreifer wählt einen Klartext  $m$ .
- Für alle Schlüssel  $k \in K$  berechnet er  $f_k(m)$ .
- Er speichert alle Ergebnisse in einer sortierten Tabelle.
- Charakteristik der Vorbereitungsphase: viel Rechenaufwand, hoher Speicherbedarf
- Charakteristik des Angriffs: geringer Rechenaufwand, hoher Speicherbedarf

**Folgerung:** Schlüssellänge sollte mindestens 112 Bit betragen.

## Designkriterien von Blockchiffren

Claude Shannon 1949:

### Konfusion

Die Bits des Chiffretextes sollten einer Zufallsbitfolge gleichen. Die Redundanz des Klartextes wird zerstört.

### Diffusion

Jedes Klartextbit und jedes Schlüsselbit beeinflusst viele Bits des Chiffretextes.

- Die Änderung eines Eingabebits ändert im Mittel die Hälfte der Ausgabebits (Lavineneffekt).
- Bei Änderung eines Eingabebits ändert sich jedes Ausgabebit mit der Wahrscheinlichkeit 0.5.
- Kein Ausgabebit hängt linear von einem Eingabebit ab.

## Feistel-Chiffren

In der Praxis haben sich *Feistel-Chiffren* gut bewährt:

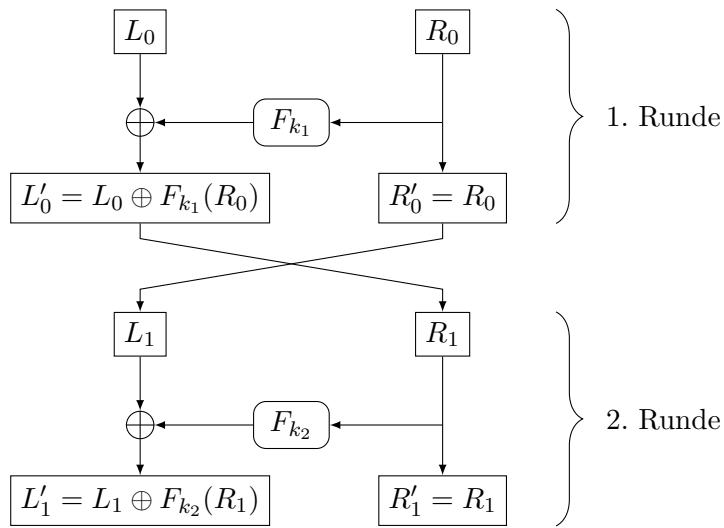
1. Der Klartext  $m$  wird in 2 Hälften  $(L_0, R_0)$  zerlegt.
2. In der  $i$ -ten Runde berechnet man

$$(L_i, R_i) = (R_{i-1}, F_{k_i}(R_{i-1}) \oplus L_{i-1})$$

3. Nach  $n$  Runden ist die Chiffre  $c = (L_n, R_n)$ .

$(k_i$  ist der Rundenschlüssel der  $i$ -ten Runde, abgeleitet aus dem Schlüssel  $k$ .)

### Ablauf einer Runde einer Feistel-Chiffre



### Feistel-Chiffren

#### Eigenschaften

- Das Ver- und Entschlüsseln geschieht mit dem gleichen Algorithmus, es müssen lediglich die Rundenschlüssel in umgekehrter Reihenfolge angewendet werden.
- $F$  wird immer nur auf die halbe Blocklänge angewendet.
- Die Sicherheit hängt im Wesentlichen von der Rundenfunktion  $F$  ab.
- Der wichtigste Vertreter einer Feistel-Chiffre ist der DES.
  - Aber auch: Blowfish, Twofish, Serpent, MARS

### Data Encryption Standard (DES)

#### Geschichte

Mai 1973	NIST startet Ausschreibung für einen Verschlüsselungs-Standard
März 1975	IBM & NSA reichen einen Kandidaten ein.
Juni 1977	DES wird veröffentlicht.
1982-1986	DES hält Einzug in Standards für Finanztransaktionen.
1993	DES wird neu zertifiziert (bis Ende der 90er)
1994	DES erstmals geknackt: 12 HP-9735-Workstations
1997	NIST kündigt Ausschreibung zum AES an.
1998	DES geknackt in weniger als 3 Tagen: 1536 Spezialchips
1999	DES geknackt in 22 Stunden: 100000 PCs
2006	COPACOBANA: 120 FPGAs zum Knacken von DES in weniger als 7 Tagen.

DES ist der erste Kryptographie-Algorithmus, der Kerckhoffs-Prinzip erfüllte.

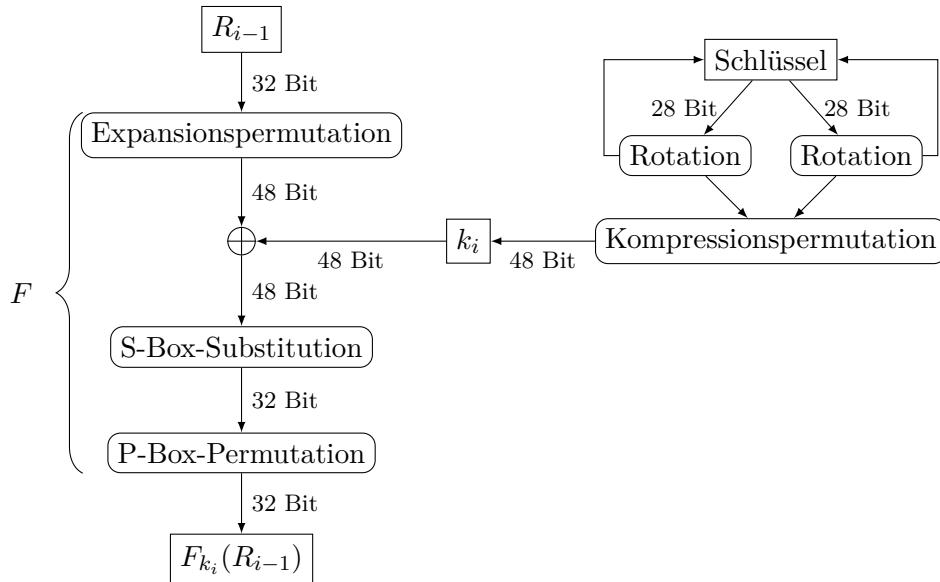
DES ist eine Feistel-Chiffre

1. mit einer Blocklänge von 64 Bit,
2. einer Schlüssellänge von 56 Bit,
3. über 16 Runden, mit einer Eingangs- und Ausgangspermutation, die invers zueinander sind.

Eine Runde des DES wird beschrieben durch

$$(L_i, R_i) = (R_{i-1}, F_{k_i}(R_{i-1}) \oplus L_{i-1}),$$

d.h. es ist noch zu klären, wie die Rundenfunktion  $F_{k_i}$  aussieht und wie die Rundenschlüssel  $k_i$  aus dem 56-Bit Schlüssel  $k$  abgeleitet werden.



Die einzige nichtlineare Operation ist die S-Box-Substitution:

1. 48-Bit werden in 8 Blöcke zu 6 Bit aufgeteilt.
2. Jeder dieser 8 Blöcke wird einer von 8 S-Boxen zugeführt.
3. Die Bits 1 und 6 werden als Zahl interpretiert und bestimmen die Zeile.
4. Die Bits 2 bis 5 werden als Zahl interpretiert und bestimmen die Spalte.
5. Die Ausgabe ist eine 4-Bit-Zahl in der entsprechenden Zeile und Spalte.

	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
6. Beispiel S-Box 5:	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

7. Angenommen die Eingabe von S-Box 5 ist 110100. Wie lautet die Ausgabe?

## Sicherheit

DES gilt als unsicher.

## Triple-DES / 3DES

Der DES wird heute noch oft als Triple-DES eingesetzt:

- Mit 3 unabhängigen Schlüsseln (Schlüsselraum umfasst 168 Bits):

$$f_{k_1,k_2,k_3}(m) = E_{k_1}(D_{k_2}(E_{k_3}(m))).$$

- Mit 2 unabhängigen Schlüsseln (Schlüsselraum umfasst 112 Bits):

$$f_{k_1,k_2}(m) = E_{k_1}(D_{k_2}(E_{k_1}(m))).$$

Frage: Warum verwendet man nicht einfach  $f_{k_1,k_2}(m) = E_{k_1}(E_{k_2}(m))$  ?

## Advanced-Encryption-Standard (AES)

### Geschichte

September 1997	NIST startet Aufruf zur Einreichung von Kandidaten
August 1998	NIST gibt 15 Kandidaten bekannt und bittet um Kommentare
April 1999	Finalkandidaten: MARS, RC6, Rijndael, Serpent, Twofish
April 2000	AES Candidate Conference
Oktober 2000	Bekanntgabe des Gewinners Rijndael
Dezember 2001	AES wird zum „federal information processing standard“ (FIPS 197).

Kerckhoffs-Prinzip wurde voll erfüllt.

Abbildung 29: Vergleich der AES-Kandidaten

	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
Total	16	14	13	10	9

## **Advanced-Encryption-Standard (AES)**

### **Anforderungen des NIST an AES**

- Blockchiffre mit Blockgröße von 128 Bit
- Schlüssellängen: 128, 192, 256 Bit
- Nachweis der Sicherheit
- einfaches Design
- AES soll effizienter als Triple-DES sein.
- effiziente Realisierung in Hard- und Software

## **Rijndael / AES**

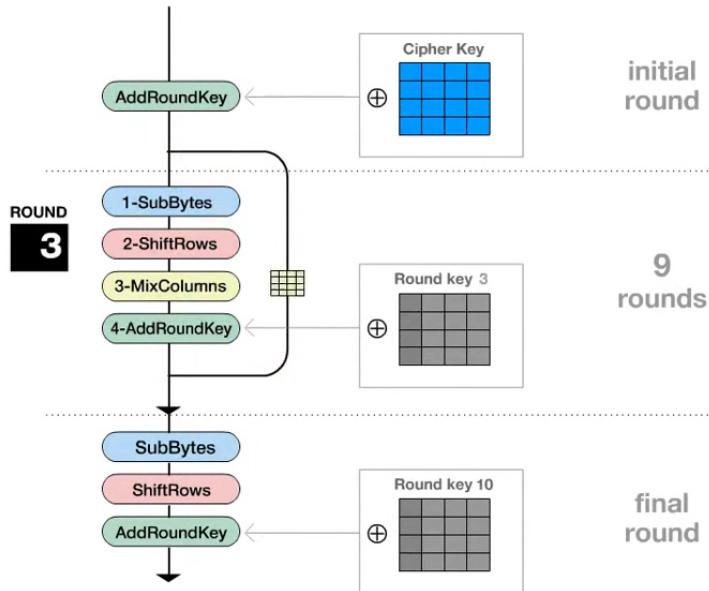
### **Eigenschaften**

- AES ist keine Feistel-Chiffre
- Die Zahl der Runden zur Verschlüsselung eines Blocks (128-Bit) ist abhängig von der Schlüssellänge:
  - 10 Runden bei 128-Bit Schlüssel,
  - 12 Runden bei 192-Bit Schlüssel,
  - 14 Runden bei 256-Bit Schlüssel
- Die Zwischenergebnisse nach jeder Runde werden Zustand genannt. Ein Zustand besteht aus einer  $(4 \times 4)$ -Matrix aus Bytes (128 Bit Blockgröße). Der initiale Zustand ist der Klartext.

## **Rijndael / AES – Ablauf des Algorithmus**

1. **Schlüsselexpansion:** Von einem AES-Schlüssel werden Rundenschlüssel abgeleitet.
2. **Vorrunde:**
  - a) **AddRoundKey:** Klartext/Zustand wird mit dem Rundenschlüssel XORed.
3. **Ablauf einer Runde:**
  - a) **SubBytes:** Substitution von Bytes durch eine S-Box
  - b) **ShiftRows:** Jede Zeile der Zustandsmatrix wird rotiert.
  - c) **MixColumns:** Jede Spalte der Zustandsmatrix wird mit einer Matrix multipliziert.
  - d) **AddRoundKey:** Zustand wird XORed mit einem Rundenschlüssel
4. **Schlussrunde:**
  - a) **SubBytes:** Substitution von Bytes durch eine S-Box
  - b) **ShiftRows:** Jede Zeile der Zustandsmatrix wird rotiert.
  - c) **AddRoundKey:** Zustand wird XORed mit einem Rundenschlüssel

Abbildung 30: Rijndael / AES – Ablauf einer Runde



### Exkurs: Rechnen auf Bytes

Moderne Blockchiffren rechnen aus Effizienzgründen auf Bytes. 1 Byte umfasst  $2^8 = 256$  Elemente, allerdings gibt es in  $\mathbb{Z}_{2^8}$  nicht immer ein multiplikatives Inverses.

Deshalb rechnet man anders ( $\mathbb{Z}_{2^8}[X]$ ): Das Rechnen wird über Polynome definiert

$$b(x) = b_7x^7 + b_6x^6 + \dots + b_1x + b_0,$$

mit den Koeffizienten  $b_i \in \{0, 1\}$ . Es gilt hier

$\oplus$	0	1
0	0	1
1	1	0

$\odot$	0	1
0	0	0
1	0	1

Das Byte 0x57=01010111 repräsentiert das Polynom  $x^6 + x^4 + x^2 + x + 1$ .

Im AES-Algorithmus wird die Multiplikation modulo  $x^8 + x^4 + x^3 + x + 1$  gerechnet.

### Exkurs: Polynome mit Koeffizienten aus Polynomen

Die Elemente aus  $\mathbb{Z}_{2^8}[X]$  werden repräsentiert durch Bytes. Wir betrachten nun Polynome vom Grad  $\leq 3$  mit Koeffizienten aus  $\mathbb{Z}_{2^8}[X]$ .

Eine 32-Bit Zahl 0x5700D411 repräsentiert so das Polynom  $0x57x^3 + 0xD4x + 0x11$ .

In der Chiffre AES werden 2 Polynome vom Grad  $\leq 3$  multipliziert und anschließend reduziert modulo  $x^4 + 1$ . Man kann zeigen, dass die Multiplikation mit  $x$  und anschließender Reduktion eine zyklische Linksverschiebung jedes Koeffizienten bewirkt. Z.B. ist

$$((0x57x^3 + 0xD4x + 0x11) \otimes x) \bmod (x^4 + 0x01) = 0xD4x^2 + 0x11x + 0x57$$

Das in AES gewählte Polynom für die Multiplikation ist  $0x03x^3 + 0x01x^2 + 0x01x + 0x02$ . Es ist teilerfremd zu  $x^4 + 1$ , somit ist die Operation umkehrbar.

## Rijndael / AES

### AddRoundKey

Aus dem initialen Schlüssel (z.B. 128 Bit) werden mehrere Rundenschlüssel abgeleitet (Rotation, XOR, S-Box).

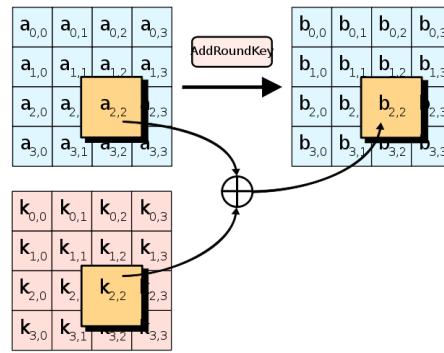


Abbildung 31: In jeder Runde wird der aktuelle Zustand mit dem Rundenschlüssel XORed.

### SubBytes / S-Box

Die Bytes des aktuellen Zustands werden mit Hilfe einer Matrix substituiert.

Mathematisch wird zunächst das multiplikative Inverse in  $\mathbb{F}_{2^8}[X]$  modulo  $x^8 + x^4 + x^3 + x + 1$  berechnet.

Und dann mit einer affinen Transformation abgebildet

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

In der Praxis erstellt man eine Tabelle und arbeitet man mit einer S-Box.

### ShiftRows

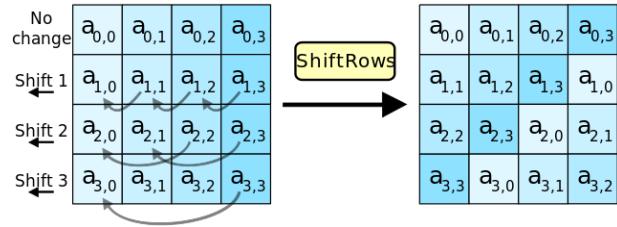


Abbildung 32: Zeilen der Zustandsmatrix werden nach links rotiert.

### MixColumns

Eine Spalte  $(a_0, a_1, a_2, a_3)$  des aktuellen Status wird elementweise in  $\mathbb{F}_{2^8}[X]$  modulo  $x^8 + x^4 + x^3 + x + 1$  mit 1, 2, 3 multipliziert.

$$\begin{aligned} b_0 &= (a_0 \cdot 2) \oplus (a_1 \cdot 3) \oplus (a_2 \cdot 1) \oplus (a_3 \cdot 1) \\ b_1 &= (a_0 \cdot 1) \oplus (a_1 \cdot 2) \oplus (a_2 \cdot 3) \oplus (a_3 \cdot 1) \\ b_2 &= (a_0 \cdot 1) \oplus (a_1 \cdot 1) \oplus (a_2 \cdot 2) \oplus (a_3 \cdot 3) \\ b_3 &= (a_0 \cdot 3) \oplus (a_1 \cdot 1) \oplus (a_2 \cdot 1) \oplus (a_3 \cdot 2) \end{aligned}$$

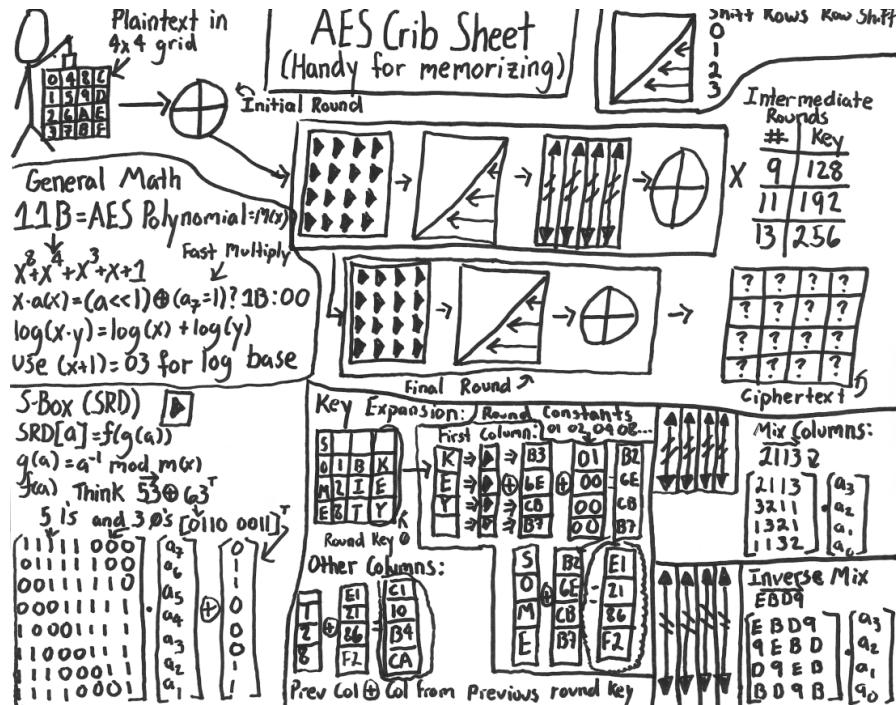


Abbildung 33: Quelle: [Moserware](#)

[Animation des AES](#), Flash-Animation by Enrique Zabala, Universität ORT, Montevideo, Uruguay.

### Eigenschaften

- **Entschlüsselung:** Jede einzelne Operation ist invertierbar.
- **Geschwindigkeit:** AES ist um ein Vielfaches schneller als DES.
- **Sicherheit:** Es existieren theoretische Angriffe, die nicht praktikabel sind.
- **Einsatz:** Verschlüsselungsfunktion, MAC, Einweg-Hash-Funktion, Stromchiffre, PRNG

### Betriebsmodi von Blockchiffren – ECB

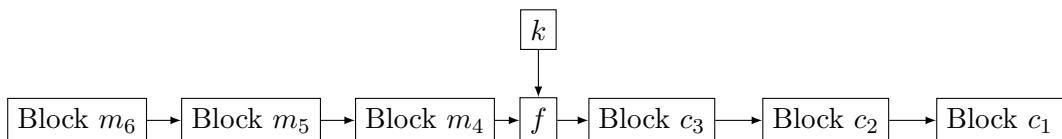


Abbildung 34: Electronic-Codebook-Modus (ECB)

- $c_i = f_k(m_i)$
- Gleiche Blöcke werden gleich verschlüsselt, z.B. Problem bei Passwortlisten.

- Blöcke können vertauscht, gelöscht oder hinzugefügt werden.
- Einzelzugriff ist möglich.
- Ein Bitfehler zerstört den gesamten Block.
- Es müssen immer ganze Blöcke verschlüsselt werden.

### Betriebsmodi von Blockchiffren – ECB

Im ECB-Modus werden gleiche Blöcke gleich verschlüsselt.



Abbildung 35: Quelle: StackExchange – Why shouldn't I use ECB encryption?

Muster des verschlüsselten Bildes lassen Rückschlüsse auf das Originalbild zu.

### Betriebsmodi von Blockchiffren – CBC

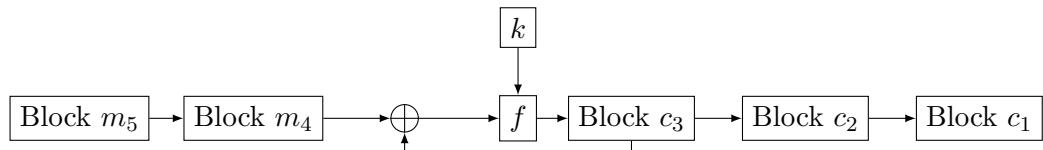


Abbildung 36: Cipher-Block-Chaining-Modus (CBC)

- $c_i = f_k(m_i \oplus c_{i-1})$ , wobei  $c_0 = IV$ .
- $IV$  soll zufällig sein und ist kein Geheimnis.
- $c_i$  hängt von allen vorangegangen Geheimtextblöcken ab.  $\Rightarrow$  Blöcke können nicht vertauscht, gelöscht oder hinzugefügt werden.
- Ein Bitfehler im Block  $c_i$  zerstört  $m_i$  und 1 Bit im Block  $m_{i+1}$ .
- Ist  $c_i = c_j$ , so folgt  $m_i \oplus c_{i-1} = m_j \oplus c_{j-1}$  bzw.  $m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$ .

### Betriebsmodi von Blockchiffren – CFB