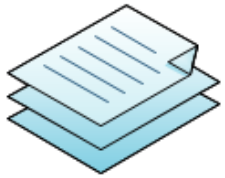
Technische Universität Berlin

7. Verschlüsselung

Jörg Schneider | Grundlagen der Rechnersicherheit | Sommersemester 2015

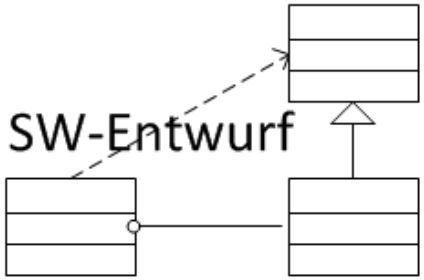
Sicherheits-
konzept



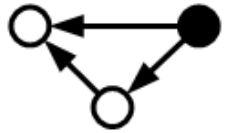
Risiko

Bedrohungen

§



<s,p,o> ϵ Z



Security | **Safety** | **Schutz** | **Sicherheit**

Wir machen keine Kryptovorlesung …

… das übernehmen die Kollegen:

- FG Security in Telecommunications (Prof. Seifert)
  - Cryptography for Security
- FG KANT Group (Algebra and Number Theory, Prof. Pohst)
  - Kryptographie
  - Codierungstheorie
- Weitere Kurse an HU und FU

Auch hilfreich Algebra und diskrete Mathematik.

3.1 Allgemeines

- Verschlüsselung zur Sicherung der Vertraulichkeit einer Nachricht wird bereits seit zwei Jahrtausenden eingesetzt.
- Mit Hilfe einer Verschlüsselungsfunktion V (Encrypt) wird der Klartext (plain text) M als Zeichenkette in eine andere Zeichenkette (ciphertext) V(M) transformiert.
- Man muss die Inverse von V als Entschlüsselungsfunktion (Decrypt) $E = V^{-1}$ kennen, um aus dem transformierten, also verschlüsselten Text V(M) den Klartext wiederherzustellen:

$$E(V(M)) = V^{-1}(V(M)) = M$$

Einsatzgebiete

Sicherung der Vertraulichkeit übertragener Information (klassisches Einsatzgebiet)
Sicherung der Vertraulichkeit gespeicherter Information (klassisches Einsatzgebiet)
Prüfung der Authentizität von Personen (digitale Unterschrift)
Prüfung der Unverfälschtheit einer übertragenen Nachricht (Message Authentication Code, MAC)
Prüfung der Authentizität von Softwarekomponenten (Zertifikate)

Technische Universität Berlin

3.2 Einfache Verfahren

Bei monoalphabetischen Substitutionen wird jeder Buchstabe des Alphabets A durch ein bestimmtes Zeichen desselben oder eines anderen Alphabets ersetzt.

Im Falle unseres Alphabets mit 26 Zeichen gibt es 26! verschiedene Codierungen.

Cäsar soll angeblich eine Substitution verwendet haben, bei der jeder Buchstabe durch seinen i-ten Nachfolger im Alphabet ersetzt wird. (Shift-Chiffre)

Beispiel (i=3, Cäsar-Chiffre)

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
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

Shift-Chiffren sowie alle anderen Formen der monoalpha-betischen Substitution können (bei natürlichsprachlichen Texten) durch einfache Häufigkeitsanalysen gebrochen werden.

Technische Universität Berlin

Buchstabenhäufigkeiten

character	English	German	character	English	German
a	8.04%	6.47%	n	7.09%	9.84%
b	1.54%	1.93%	o	7.60%	2.98%
c	3.06%	2.68%	p	2.00%	0.96%
d	3.99%	4.83%	q	0.11%	0.02%
e	12.51%	17.48%	r	6.12%	7.54%
f	2.30%	1.65%	s	6.54%	6.83%
9	1.96%	3.06%	t	9.25%	6.13%
h	5.49%	4.23%	u	2.71%	4.17%
i	7.26%	7.73%	v	0.99%	0.94%
j	0.16%	0.27%	w	1.92%	1.48%
k	0.61%	1.46%	x	0.19%	0.04%
l	4.14%	3.49%	y	1.73%	0.08%
m	2.53%	2.58%	z	0.09%	1.14%

Technische Universität Berlin

Polyalphabetische Verfahren

Bei polyalphabetischen Verfahren wird ein Zeichen nicht immer auf dasselbe abgebildet.

Bei den Vigenère-Verfahren wird ein Schlüssel verwendet, der Zeichen für Zeichen mit dem Klartext verknüpft wird.

Beispiel: Der Schlüssel ist „Alice“ und er wird (wenn nötig wiederholt) unter den Klartext geschrieben. Die jeweils übereinanderstehenden Zeichen werden additiv verknüpft.

```
ALLESISTEINEFOLGEVONBITS
ALICEALICEALICEALICEALIC
------------------------
AWTGWIDBGMNPNQPGPDQRBTBU
```

Addition (A=0, B=1,...)

Die Entschlüsselung erfolgt analog durch Subtraktion des Schlüssels (modulo 26).

Vernam-Chiffre

Einen Spezialfall der Vigenère-Verfahren bildet die Vernam-Chiffre. Dabei muss der Schlüssel so lang sein wie der Klartext.

Man kann z.B. einen Teil eines Buches als Schlüssel verwenden.

Auch hier kann man mit Häufigkeitsanalysen erfolgreich angreifen.

Besseren Schutz bilden die One-time-pads, bei denen der Schlüssel kein natürlichsprachlicher Text, sondern eine zufällige Zeichenfolge ist, die auch nur einmal verwendet wird.

Üblicherweise wird der One-time-pad auf Bitfolgen angewendet mit XOR als Verknüpfung.

One-time-pads sind zwar sicher, aber für die meisten Anwendungsfälle unpraktikabel.

Verschlüsselung		Entschlüsselung	
0111010100010	Klartext	1011110111001	Chiffretext
1100100011011	One-time-pad	1100100011011	One-time-pad
	XOR		XOR
1011110111001	Chiffretext	0111010100010	Klartext

Permutationsverfahren

Bei diesen Verfahren werden die Buchstaben nicht durch andere ersetzt, sondern untereinander permutiert.

Man fasst jeweils *n* Buchstaben zusammen und permutiert sie nach einer festen Abbildung:

Beispiel $n = 4$, Permutation $\pi = (3,1,4,2)$

PERMUTATIONSVERFAHREN wird zu

RPMEAUTTNISORVFERAEHN

Entschlüsselung geschieht analog durch die Inverse π^{-1}

Im Beispiel ist die Inverse $\pi^{-1} = (2,4,1,3)$.

Technische Universität Berlin

3.3 Verschlüsselung heute

Mit dem Aufkommen von Rechnern und deren Analysemöglichkeiten sind die Anforderungen an kryptographische Verfahren extrem gestiegen.

Auch bei Einsatz von Hochleistungsrechnern soll es in akzeptabler Zeit nicht möglich sein:

- bei Kenntnis von V(M) die Nachricht M zu ermitteln,
- bei Kenntnis einer codierten Nachricht V(M) und des Klartextes M die Verschlüsselungsfunktion V bzw. ihre Inverse E zu ermitteln.

Untauglich sind daher z.B. alle Verfahren, die einzelne Zeichen oder Buchstaben verschlüsseln:

Natürlichsprachige Texte hinreichender Länge können durch Häufigkeitsanalyse der Buchstaben leicht entschlüsselt werden.

Das gleiche trifft auch auf Buchstabenpaare (Digramme) oder -tripel (Trigramme) zu.

Angriffsarten

Ciphertext-only-attack

Der Angreifer kennt lediglich den verschlüsselten Text (Abfangen oder Mithören verschlüsselter Nachrichten).

Known-plaintext-attack

Der Angreifer kennt sowohl den Chiffretext, als auch den Klartext (Kenntnis z.B. von Teilen der Nachricht: Inhalt des Email-Headers).

Chosen-plaintext-attack

Der Angreifer kann den Klartext selbst wählen (und kann ihm bestimmte statistische Eigenschaften geben).

Die Schwierigkeit des Angriffs nimmt von oben nach unten ab.

Gute Kryptoverfahren müssen auch Chosen-plaintext-Angriffen widerstehen.

Immer möglich:
Brute Force, d.h. alle Schlüssel ausprobieren

Allgemeine Anforderungen

Blocklänge:

Die Blocklänge (Einheit der Verschlüsselung) sollte groß genug sein, um Häufigkeitsanalysen unwirksam werden zu lassen.
64/128-bit-Blöcke werden als hinreichend lang angesehen.

Schlüssellänge

Der Schlüssel sollte lang genug sein, um ein vollständiges Ausprobieren aller Schlüssel unmöglich zu machen. Die sinnvolle Schlüssellänge hängt vom Verfahren ab.

Konfusion

Der verschlüsselte Text sollte in möglichst komplizierter Weise von der Kombination von Klartext und Schlüssel abhängen. Keine statistischen Zusammenhänge.

Diffusion

Jedes Bit des Klartextes sollte jedes Bit des verschlüsselten Texts beeinflussen. Ebenso sollte jedes Bit des Schlüssels jedes Bit des verschlüsselten Texts beeinflussen.

Maximen

Keiner sollte den Angreifer unterschätzen.

Nur ein Kryptoanalytiker – wenn überhaupt - kann die Sicherheit eines Kryptoverfahrens beurteilen.

Bei der Beurteilung der Sicherheit eines Kryptosystems ist davon auszugehen, dass der Angreifer den Algorithmus kennt. (Nur den Schlüssel nicht.) (Kerckhoffs Prinzip)

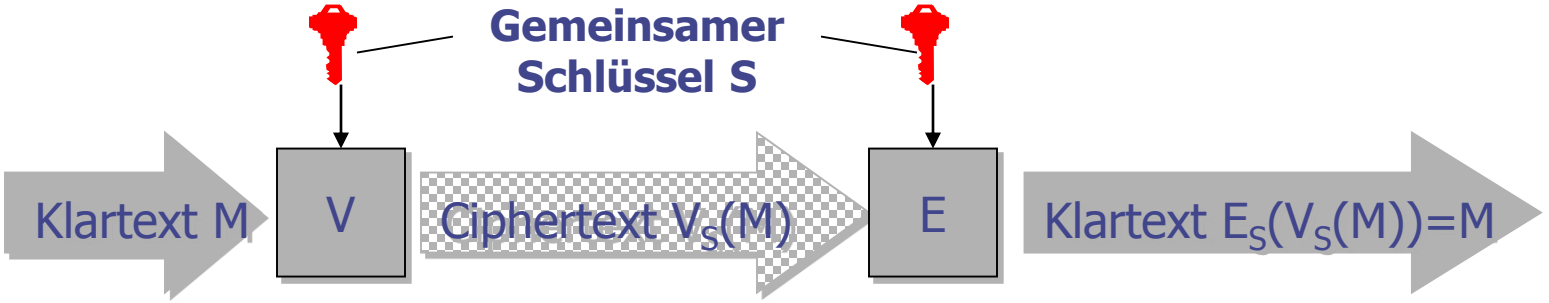
Zusätzliche Komplikationen machen ein Verfahren nicht unbedingt sicherer.

Menschliche Fehler bei Entwurf, Implementierung und Nutzung von Kryptoverfahren müssen ebenfalls berücksichtigt werden.

3.4 Symmetrische Verschlüsselung (Secret Key Encryption)

Ein Verschlüsselungsalgorithmus besteht in der Regel aus einer Familie von Verschlüsselungsfunktionen, aus denen durch Vorgabe eines speziellen Parameters, des Schlüssels, eine bestimmte ausgewählt wird.

Die Wahl des Schlüssels sollte erheblichen Einfluss auf die Ausgabe der Funktion V_S besitzen.



Der Schlüssel S muss sowohl Sender wie auch Empfänger bekannt sein.

In der Praxis eingesetzte Verfahren sind so beschaffen, dass V und E bekannt sein dürfen und lediglich der Schlüssel S geheim gehalten werden muss. Er muss daher auf anderem Wege Sender und Empfänger mitgeteilt bzw. zwischen ihnen ausgetauscht werden.

3.4.1 DES Data Encryption Standard

Nationaler Standard in den USA (früher Exportverbot)

Der Klartext (Bitfolge beliebiger Art) wird in 64 bit - Portionen zerlegt, die in 16 Phasen gesteuert durch einen 56 bit langen Schlüssel völlig durcheinander gewürfelt werden.

Der Algorithmus ist programmtechnisch relativ kompliziert und daher zeitaufwändig, kann jedoch leicht und effizient in Hardware realisiert werden.

Die Entschlüsselung verläuft fast identisch, d.h. kann von derselben Hardware durchgeführt werden.

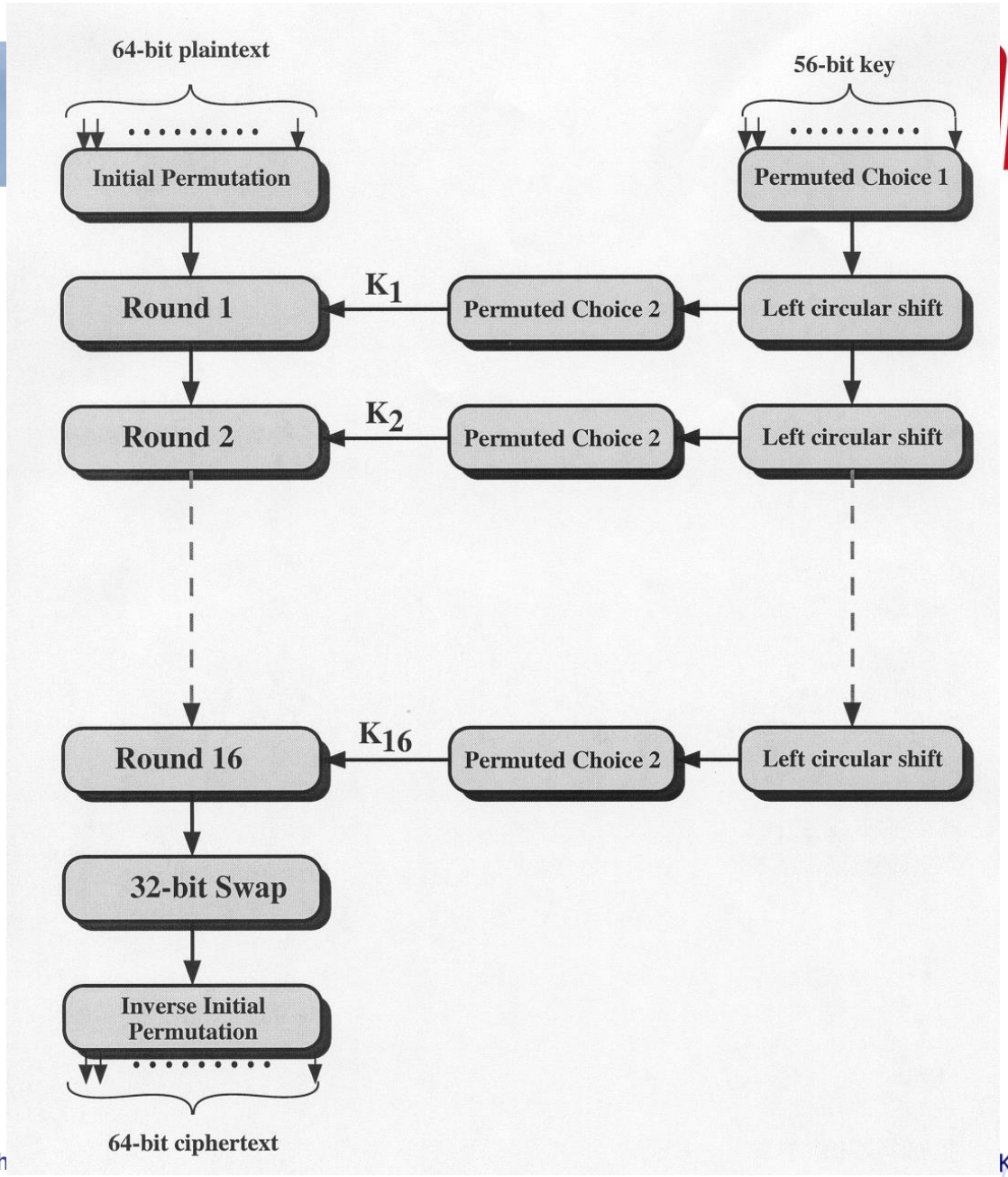
DES besitzt die wünschenswerte Eigenschaft, dass

- minimale Änderungen des Schlüssels
- minimale Änderungen des Klartextes

erhebliche Auswirkung auf den verschlüsselten Text besitzen.

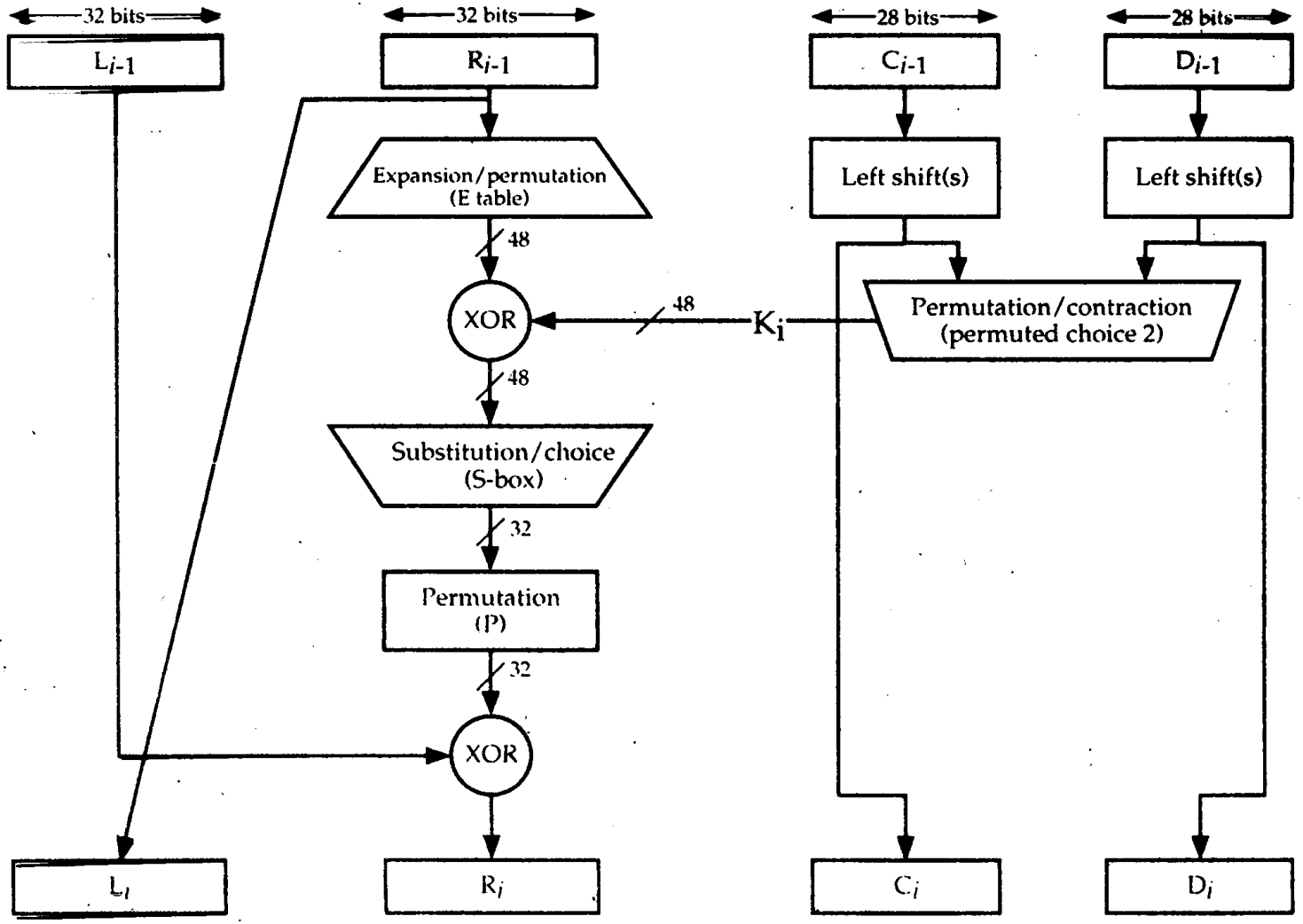
Der DES gilt heute (d.h. bei der heutigen Rechnerleistung) als nicht mehr hinreichend sicher.

DES-Details

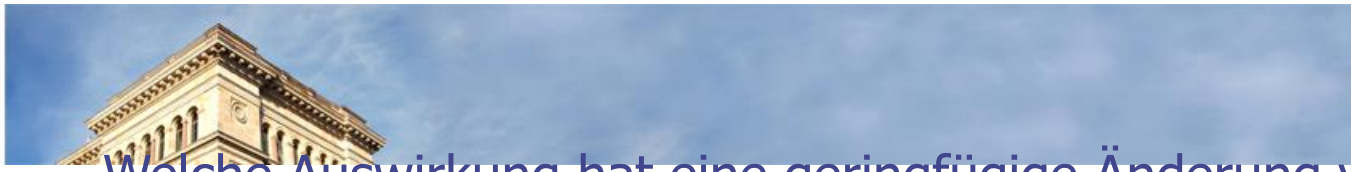


Quelle: W. Stallings

D



Quelle: W. Stallings



Technische Universität Berlin

Welche Auswirkung hat eine geringfügige Änderung von Klartext oder Schlüssel auf den Chiffretext?

„Lawineneffekt“ in DES

Modifikation eines Bits in einem 64-bit Klartext

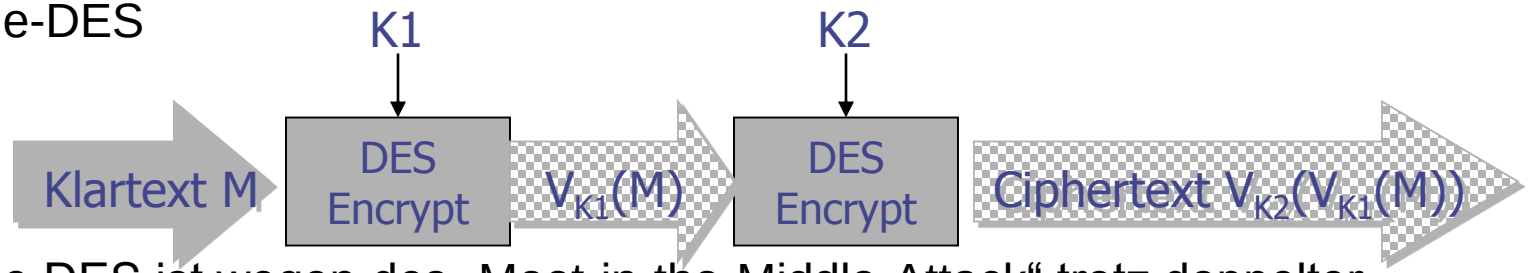
(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Modifikation eines Bits in einem 56-bit-Schlüssel

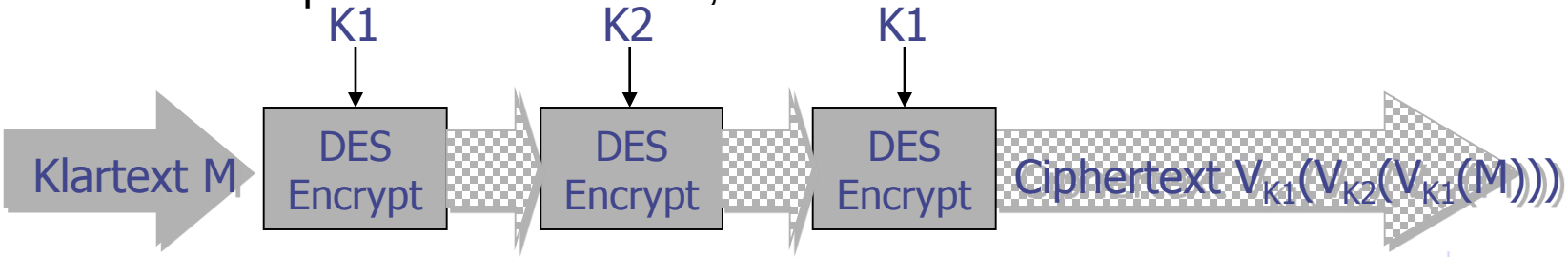
Quelle: W. Stallings

Verbesserung von DES

DES ist wegen seines kurzen Schlüssels nicht mehr sicher.
Bereits 2001 gelang ein Known-plaintext-Angriff in 22 h durch paralleles Internet-Computing.
Zur Verbesserung wird kaskadierter Einsatz vorgeschlagen:
Double-DES



Double DES ist wegen des „Meet-in-the-Middle-Attack“ trotz doppelter Schlüssellänge nur unwesentlich sicherer als einfaches DES.
Daher sollte man Triple-DES verwenden, wobei zwei Schlüssel ausreichen:



3.4.2 Advanced Encryption Standard (AES)

Offizieller Nachfolger von DES ist der AES, für dessen Funktionsweise 1998 eine öffentliche Ausschreibung stattfand. (NIST = National Institute of Standards and Technology)

Anforderungen:

- Blockverschlüsselung: 128-bit-Blöcke
- Schlüssellänge: 128/192/256 bit
- Effiziente Implementierbarkeit
- Dokumentation, C-Code, Java-Code, Testdaten
- Weltweite freie Verfügbarkeit

Aus 15 eingereichten Vorschlägen wurde in 2 Runden das beste Verfahren ausgewählt und im Oktober 2000 bekannt gegeben:

Gewinner wurde der Algorithmus „Rijndael“ der belgischen Mathematiker Daemen und Rijmen

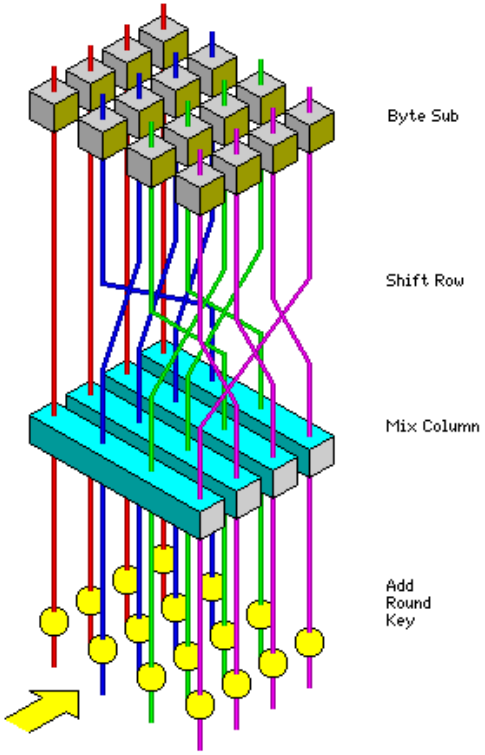
AES (Rijndael)

Blocklänge: 128/192/256
Schlüssellänge: 128/192/256
Anzahl Runden:

- 10 falls Block und Schlüssel 128 bit
- 12 falls Block oder Schlüssel 192 bit
- 14 falls Block oder Schlüssel 256 bit

In jeder Runde 4 Schritte:

(1) Byte Substitution in S-Box
(2) Shift Row
(3) Mix Column
(4) Add Round Key



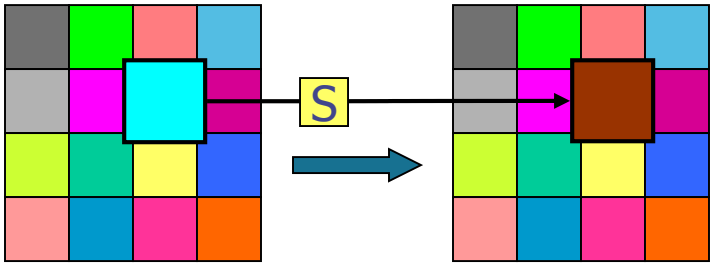
AES Details (256-Bit-Blöcke)

128-bit Block *b* als 4x4 Matrix von Bytes wird in jedem Schritt transformiert ($b \mapsto b'$)

Schritt 1 (Byte Substitution)

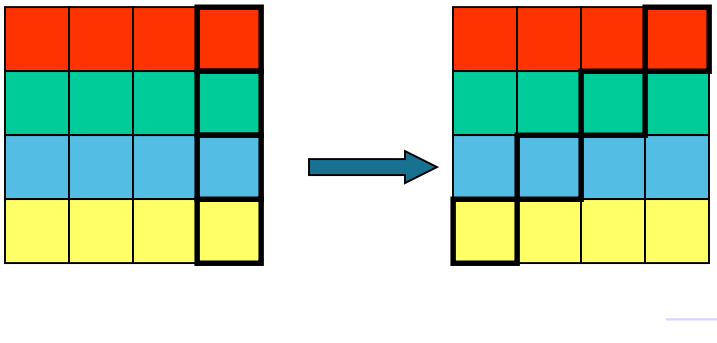
Jedes Byte wird durch eine S-Box transformiert.

(Implementiert als Tabelle)



Schritt 2 (Row Shift)

Die Zeilen werden zyklisch um 0, 1, 2 oder 3 Byte verschoben

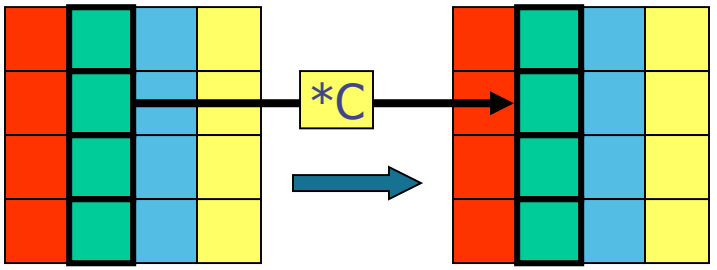


AES Details

128-bit Block als 4x4-Matrix von Bytes

Schritt 3 (Mix Column)

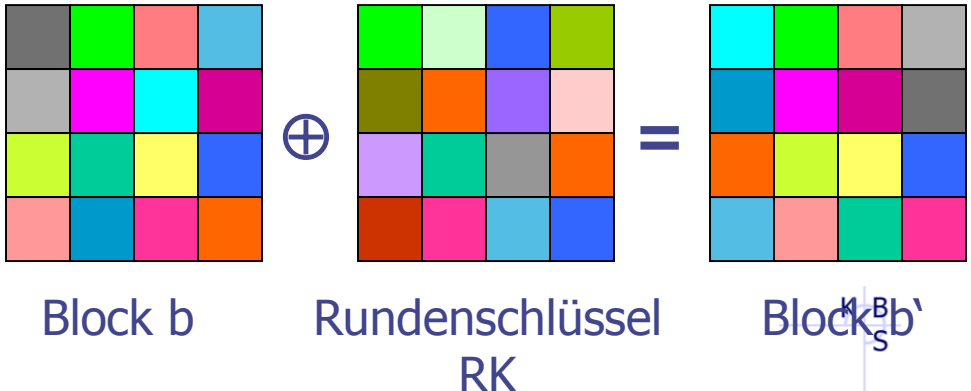
Jede Spalte wird mit einer Matrix multipliziert



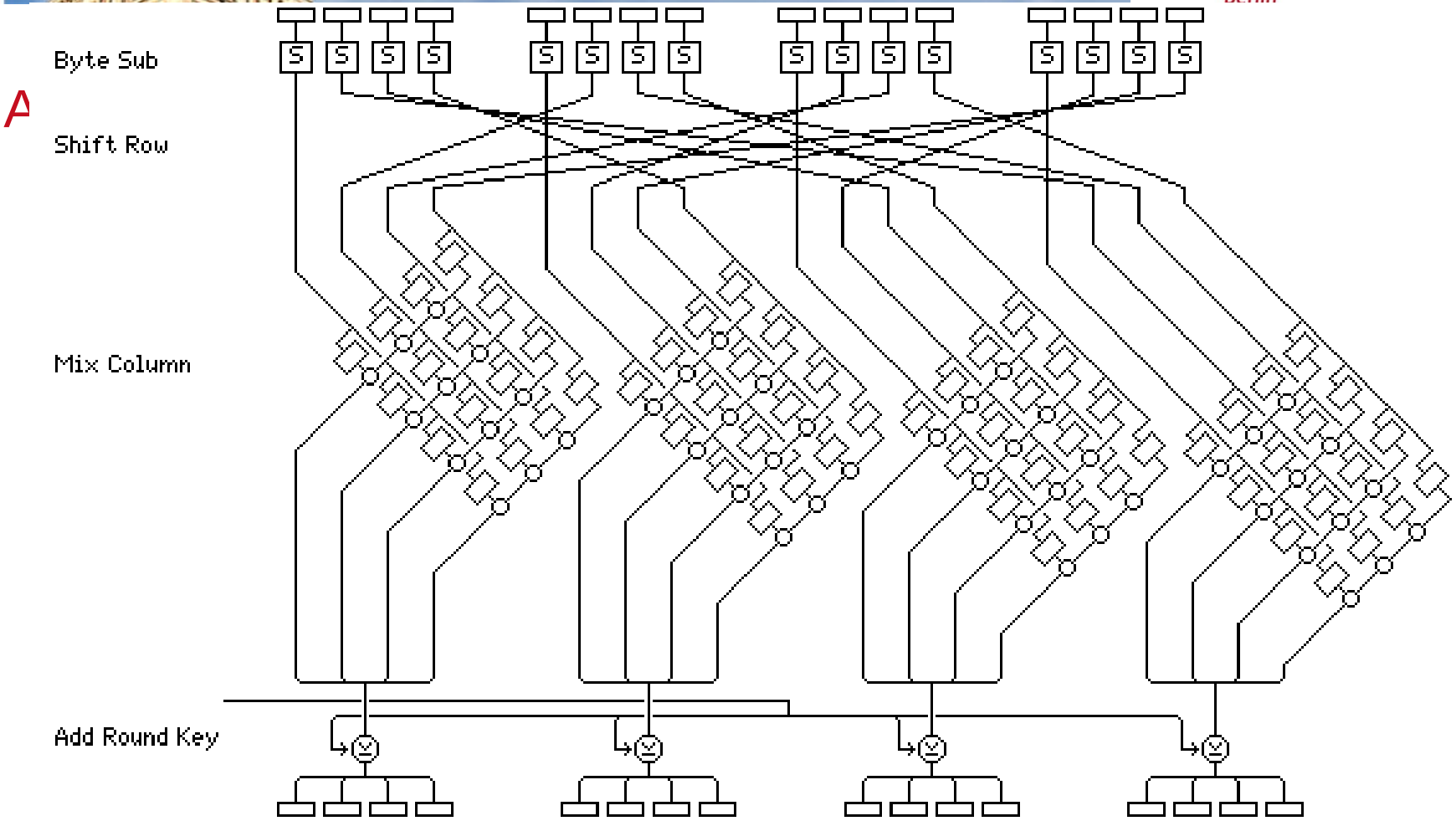
Schritt 4 (Add Round Key)

XOR-Verknüpfung mit dem Rundenschlüssel

(Die Rundenschlüssel werden aus dem Schlüssel K erzeugt)



Technische Universität Berlin



Technische Universität Berlin

Was machen wir,
wenn wir mehr Daten haben
als in einen Block passen?

K B S

Verkettung

Im einfachsten Fall (Electronic Codebook, ECB) wird jeder Block Klartext in einen Block Schlüsseltext transformiert.

Zwei identische Klartextblöcke werden dann auch auf identische Schlüsseltextblöcke abgebildet.

Bei längeren Texten vereinfacht das einen Angriff.

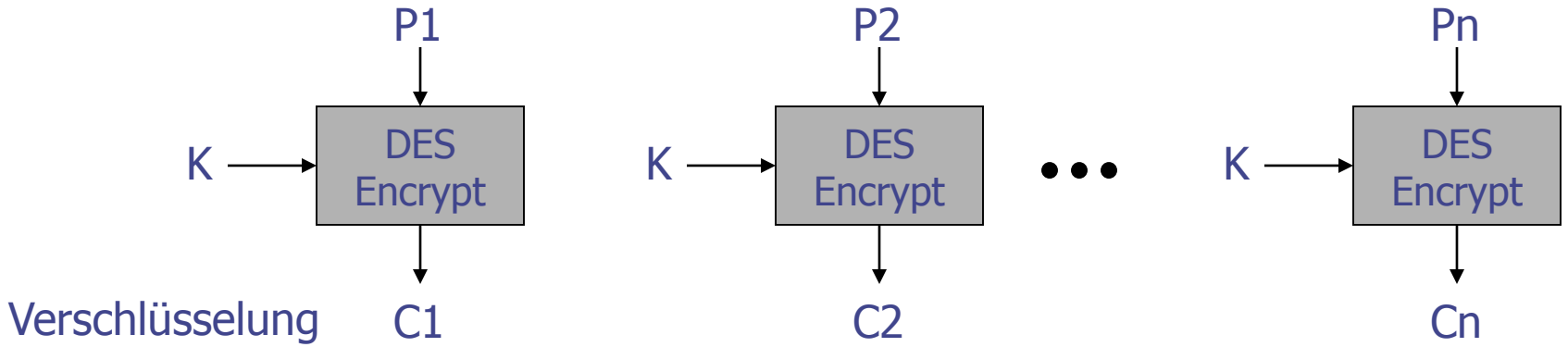
ECB eignet sich daher nur für kurze Nachrichten (z.B. Übertragung eines Schlüssels)

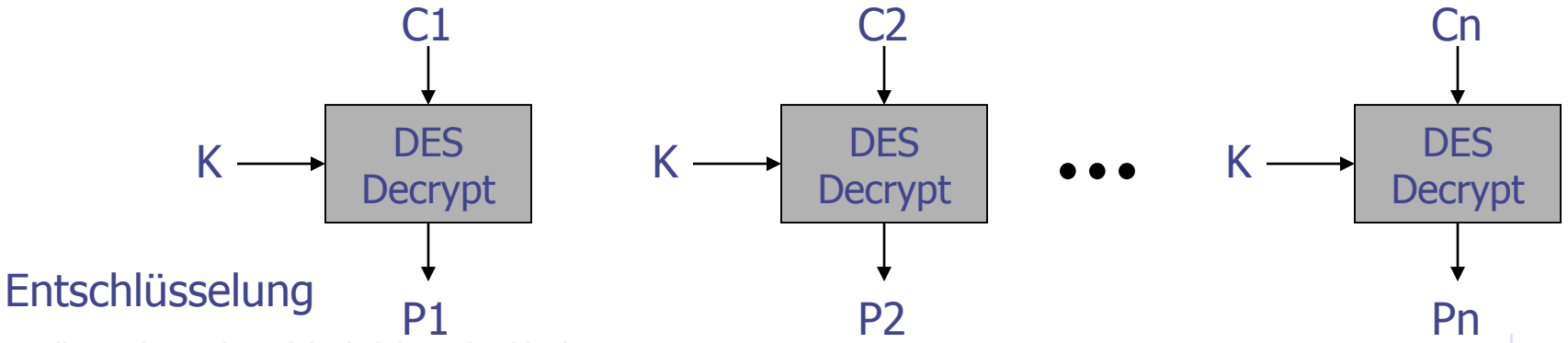
Zusätzlich gibt es Verfahren der Verkettung, bei denen der vorangegangene Block die Verschlüsselung des nachfolgenden Blocks beeinflusst: Cipher block chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB)

Bei CBC wird der nächste Klartextblock mit dem vorangegangenen Schlüsseltextblock XOR-verknüpft.

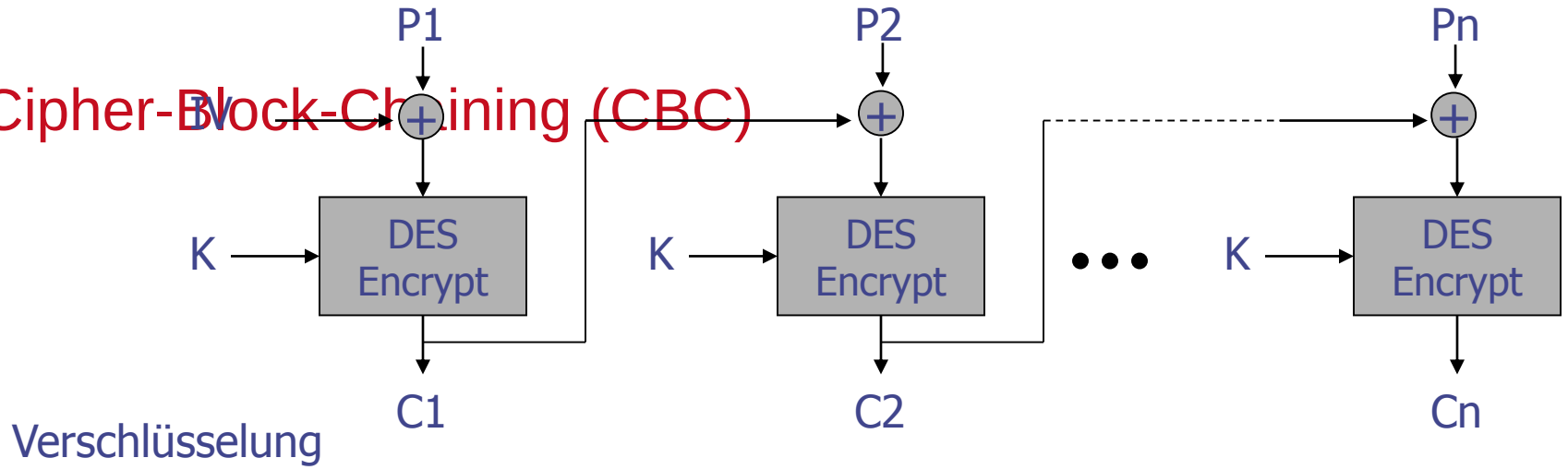
Zur Initialisierung wird ein Initialisierungsvektor (in Blockgröße) verwendet, der ebenfalls geheimzuhalten ist.

Electronic Codebook (ECB)

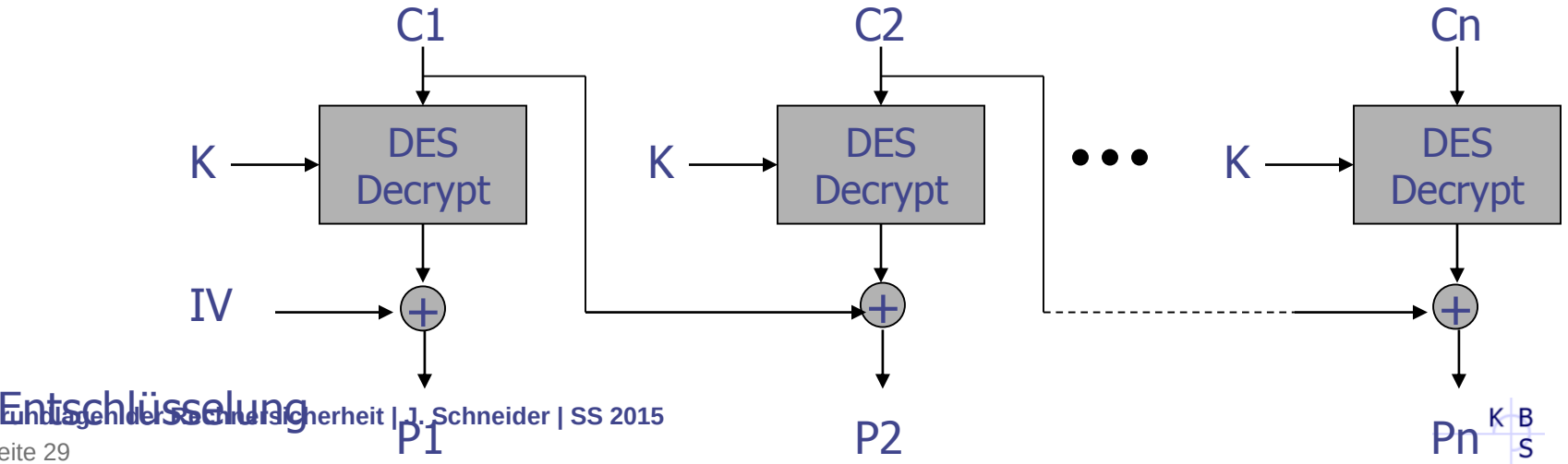




Cipher-Block-Chaining (CBC)



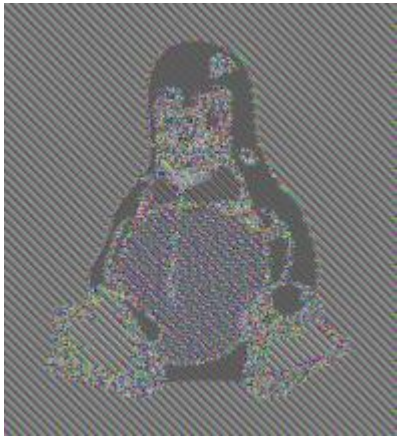
Verschlüsselung



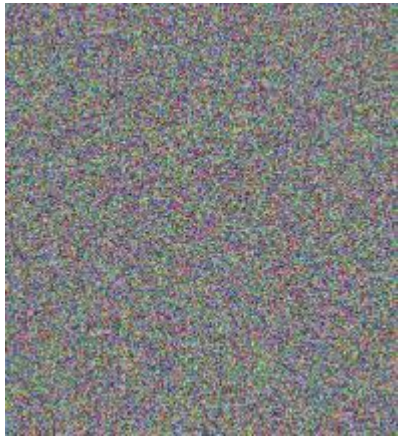
Entschlüsselung

Beispiel

Original



ECB



Andere Verkettung

http://en.wikipedia.org/wiki/Cipher_feedback
+ WikiMedia Commons

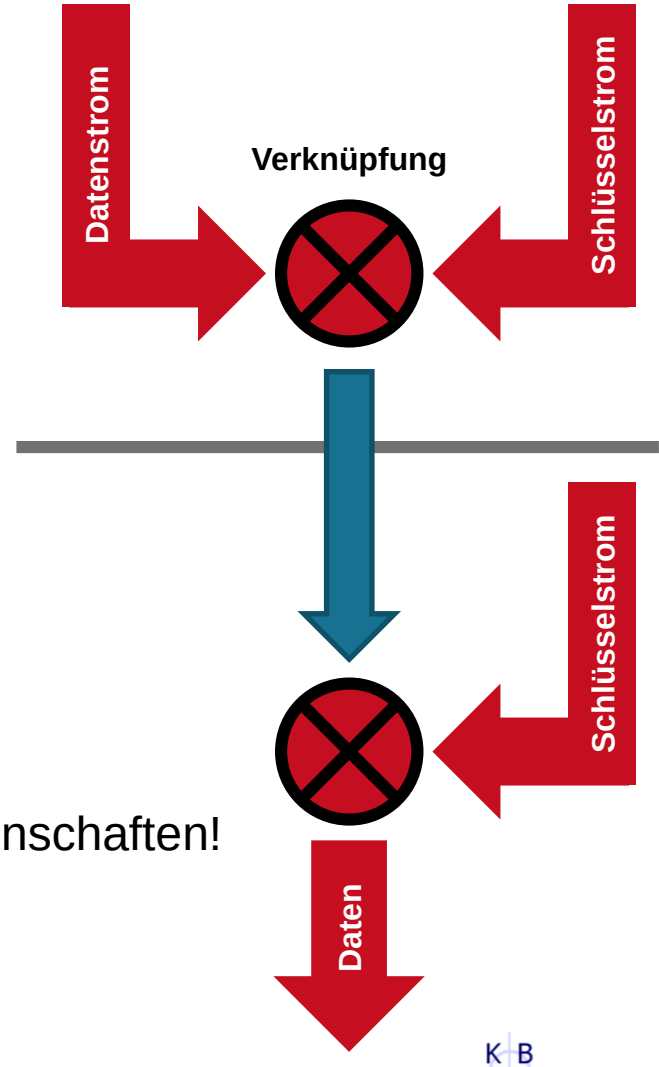
Technische Universität Berlin

Kann man nicht gleich beliebig große Daten verschlüsseln?

Stromverschlüsselung (Stream Cipher)

Idee: Es werden immer so viele Daten verschlüsselt, wie gerade vorliegen

- Angelehnt an One-Time-Pad
- Verschlüsselt kleine Einheiten (Bits oder Byte)
- Verknüpfung z.B. XOR
- Schlüsselstrom
  - Basiert auf Schlüssel, Nonce und/oder IV
  - Möglichst zufällig und lange Periode
  - Sollte nicht zweimal verwendet werden
  - Fehler bei der Generierung zerstören Eigenschaften!
- Beispiele: RC4 (SSL, WEP), A5/1+2 (GSM)



Problem des Schlüsselaustauschs

Symmetrische Verfahren sind sicher und schnell.
Sie benötigen jedoch für jede Kommunikationsverbindung einen individuellen Schlüssel.
Bei *n* Knoten, die miteinander kommunizieren wollen, werden *n* (*n* -1) / 2 Schlüssel benötigt.
Symmetrische Verfahren setzen voraus, dass der geheime Schlüssel bereits auf sicherem Wege ausgetauscht wurde.
Wie soll das gehen bei weltweiten Kommunikationen im Internet?
Neue Idee: Es gibt zwei Schlüssel: Einer davon darf öffentlich sein. (Diffie und Hellman, 1976).

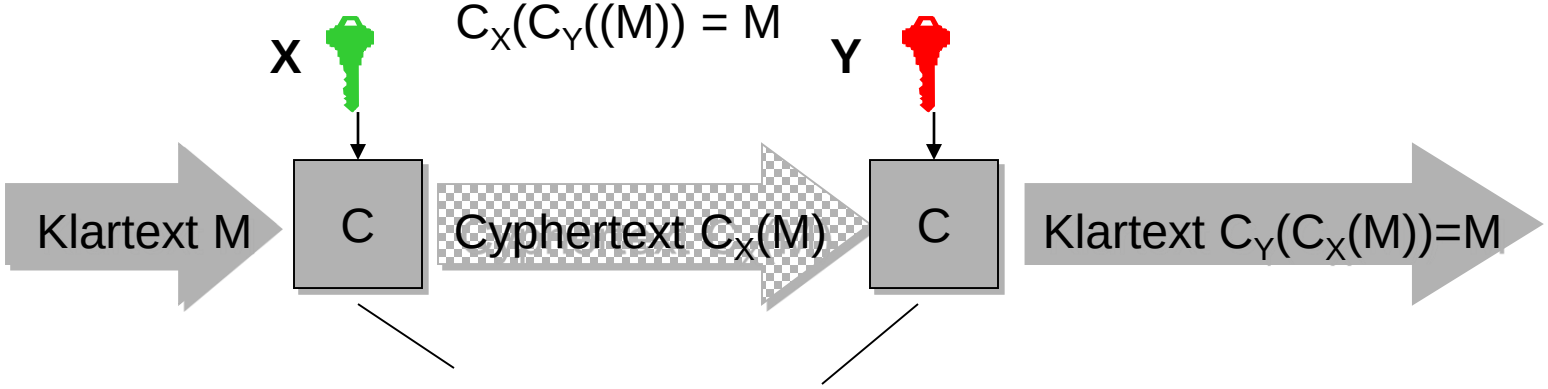
3.5 Asymmetrische Verfahren

Es werden zwei verschiedene Schlüssel verwendet

- C Codierungsfunktion (Ver- und Entschlüsselung)
- X,Y Schlüsselpaar

Es muss gelten $C_Y(C_X((M)) = M$

$C_X(C_Y((M)) = M$



Identische Funktion

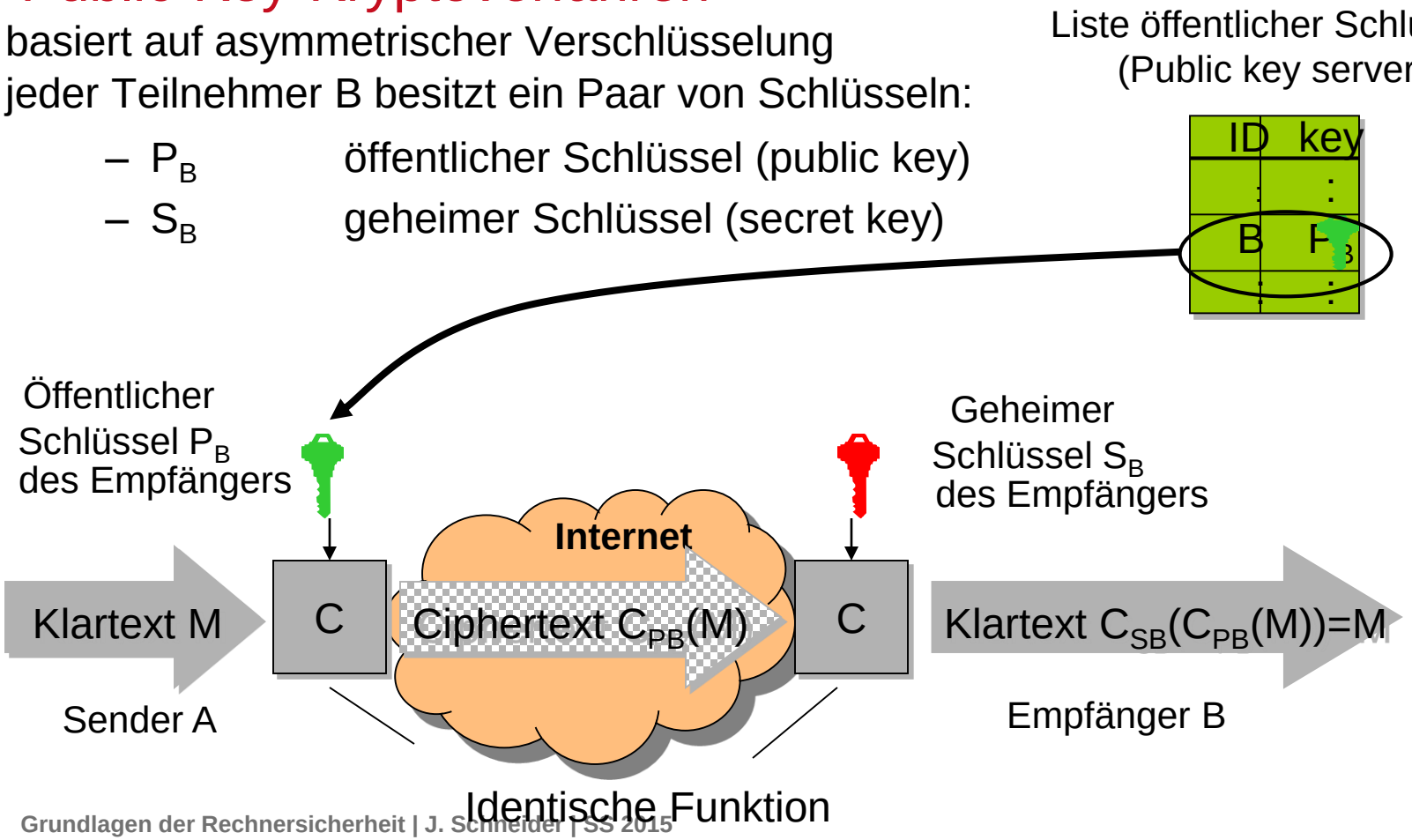
Mindestens ein Schlüssel X oder Y muss geheim sein

(z.B. RSA, ElGamal, DSS, Elliptische Kurven Krypto.)

Public-Key-Kryptoverfahren

basiert auf asymmetrischer Verschlüsselung
jeder Teilnehmer B besitzt ein Paar von Schlüsseln:

- P_B öffentlicher Schlüssel (public key)
- S_B geheimer Schlüssel (secret key)



Das RSA-Verfahren (Rivest, Shamir, Adleman, 1977)

Zahlentheoretische Vorbemerkungen

Sei $Z_n = \{0,1,2,...,n-1\}$ und $a,b \in Z_n$
Dann sei die Summe $\oplus$ und das Produkt $\otimes$ folgendermaßen definiert:

$a \oplus b = (a+b) \bmod n$

$a \otimes b = (a \cdot b) \bmod n$

Bezüglich der Addition bildet $(Z_n, \oplus)$ eine kommutative Gruppe. Zusammen mit der Multiplikation bildet $(Z_n, \oplus, \otimes)$ einen Ring, den so genannten Restklassenring.

Sei $a, b \in Z_n$. Dann heißt b mit $a \otimes b = 1 \bmod n$ das multiplikative Inverse zu a.
Ist $a \in Z_n$ ein Teiler von n, dann gibt es kein b mit $a \otimes b = 1 \bmod n$.
Daher ist Z_n keine multiplikative Gruppe.
Entfernt man jedoch alle Teiler von n aus Z_n, so erhält man die Menge Z_n^*, die dann eine multiplikative Gruppe $(Z_n^*, \otimes)$ bildet:

$Z_n^* := \{ a \in Z_n \mid ggT(a,n)=1\}$

Das RSA-Verfahren (2)

Die Zahl der Elemente von Z_n^* wird mit $\varphi(n)$ *bezeichnet* (Eulersche φ-Funktion).

$\varphi(n)$ gibt (für $n>1$) die Anzahl der positiven ganzen Zahlen an, die zu n relativ prim sind, d.h. diejenigen Zahlen m, für die gilt: $ggt(n,m) = 1$.

Es gilt: $\varphi(n) = n \prod_{p|n} (1 - 1/p)$, wobei p alle Primzahlen durchläuft, die Teiler von n sind.

Beispiel: $\varphi(45) = 45 \cdot (1 - 1/3) \cdot (1 - 1/5) = 45 \cdot 2/3 \cdot 4/5 = 24$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18	8	12

- Ist p eine Primzahl, so gilt $\varphi(p) = p - 1$
- Ist p eine Primzahl und ist $m \in \mathbb{N}$, so gilt $\varphi(p^m) = (p-1) \cdot p^{m-1}$
- Ist n ein Produkt aus zwei Primzahlen $n = p \cdot q$, so gilt $\varphi(n) = (p-1) \cdot (q-1)$

Technische Universität Berlin

Das RSA-Verfahren (3)

RSA basiert auf der Schwierigkeit, große Zahlen zu faktorisieren und verwendet den folgenden Satz von Euler-Fermat:

Sei $a \in Z_n^*$. Dann gilt $a^{\varphi(n)} \equiv 1 \pmod{n}$

Man wählt zwei Zahlen x und y, für die gilt:

$x \cdot y \equiv 1 \bmod \varphi(n)$, die also multiplikativ invers zueinander sind

Dann gilt auch

$$x \cdot y = k\,\varphi(n) + 1 \quad \text{mit } k \in \mathbb{N}$$

Sei nun $m \in Z_n^*$. Dann gilt nach obigem Satz

$$\begin{aligned}(m^x)^y &\equiv m^{k\,\varphi(n)+1} \pmod{n} \\ &\equiv (m^{\varphi(n)})^k\, m \pmod{n} \\ &\equiv 1^k\, m \pmod{n} \\ &\equiv m \pmod{n}\end{aligned}$$

Das RSA-Verfahren (4): Schlüsselgenerierung

Konstruktion eines Schlüsselpaares *X,Y:*

- Wähle (zufällig) zwei große ($> 10^{100}$), verschiedene Primzahlen *P* und *Q* und bilde $N = P \cdot Q, \quad Z = \varphi(N) = (P - 1) \cdot (Q - 1)$
- Als zufälligen Schlüssel *X* wähle eine Zahl relativ prim zu *Z*, d.h. *ggt(X,Z)* = 1

Als dazu passenden Dechiffrierschlüssel ist ein Y zu finden, für das gilt:
$X \cdot Y \equiv 1 \ mod \ Z$ oder $X \cdot Y = 1 + k\,(P - 1) \cdot (Q - 1)$ für ein $k \in \mathbb{N}$
d.h. *X* und *Y* sind reziprok zueinander (mod Z),
Y ist also der *Kehrwert* von *X* bezüglich des Moduls *Z*.

Damit ist $X \cdot Y$ die kleinste natürliche Zahl, die bei Division durch *Z* einen Rest von 1 ergibt und durch *X* teilbar ist.

Y ist dann ebenfalls relativ prim zu *N* und kann nach Wahl von X aus Z berechnet werden (oder umgekehrt). (Erweiterter Euklidscher Algorithmus)

P und *Q* sind geheim zu halten, ihr Produkt *N* ist jedoch öffentlich.

Nach der Erzeugung von *X* und *Y* wird zur Ver- und Entschlüsselung außer *X* bzw. *Y* nur noch die Zahl *N* benötigt, die dem eigentlichen Schlüssel *X* bzw. *Y* beigegeben wird.

RSA-Verfahren (5): Nutzung

Es werden also die folgenden beiden Schlüssel verwendet:

(X,N) und (Y,N)

Jeder Schlüssel besteht also aus einem Zahlenpaar. Ein Schlüssel ist geheim, der andere öffentlich.

Bezeichnet M die Nachricht im Klartext und C die verschlüsselte Nachricht, so wird folgende Verschlüsselungsfunktion V bzw. Entschlüsselungsfunktion E verwendet.

- Verschlüsselung: $C = V_{X,N}(M) = M^X \bmod N$
- Entschlüsselung: $M = V_{Y,N}(C) = C^Y \bmod N$

Es gilt: $M = C^Y \bmod N = (M^X \bmod N)^Y \bmod N \equiv M^{XY} \bmod N \equiv M$

(Beweis z.B. bei Cormen, Leiserson, Rivest: „Introduction to Algorithms“, S. 835)

Als Blocklänge wählt man k Bits, so dass k hinreichend groß ist, aber $2^k<N$ gilt.

Technische Universität Berlin

RSA-Verfahren: Beispiel

Wähle $P = 11$, $Q = 29$

Daraus folgt $N = 319$ und $Z = \varphi(N) = 280$

Wähle $X = 3$ (relativ prim zu 280)

$X\ Y = 1\ mod\ 280 = 1,\ 281,\ 561,\ 841, \ldots$

Die kleinste durch 3 teilbare dieser Zahlen ist 561, also $Y = 561 / 3 = 187$

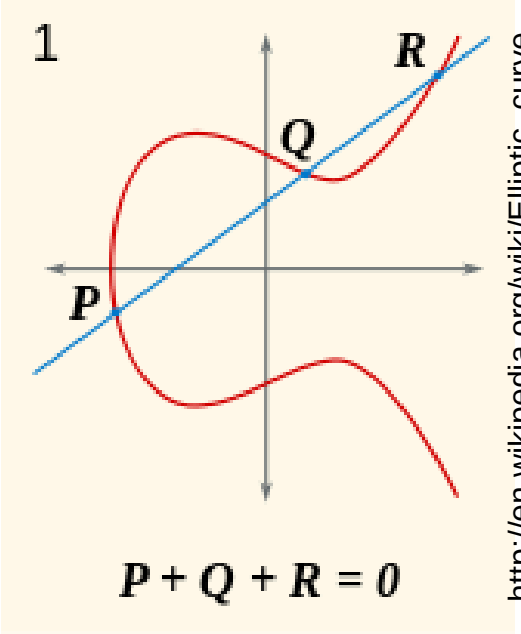
Ein Klartext $M = 5$ wird also verschlüsselt als $5^{187}\ mod\ 319 =$
509789411562384728649241727285667777891534576616380115436\43
473083572128484178611501788759910458459501114703016355631735
65673828125 *mod* 319 = 80

Die Entschlüsselung ergibt $80^3\ mod\ 319 = 512000\ mod\ 319 = 5$

Elliptische-Kurven-Kryptographie (ECC)

Ähnlich wie bei den Restklassen, kann man über Punkte auf einer elliptischen Kurve auch eine zyklische Gruppe definieren.

- Sicherheit basiert auf diskretem Logarithmus Problem
- Bei gleicher Sicherheit kürzere Schlüssel als RSA
- Vorgegebene Kurven:
  - NIST
  - BSI TR-03111
  - ECC-Brainpool / IETF RFC RFC 5639



http://en.wikipedia.org/wiki/Elliptic_curve

Elliptische-Kurven-Kryptographie (ECC)

Wenn Elliptische Kurven kürzere Schlüssel bei gleicher Sicherheit verwenden, wieso benutzt dann überhaupt noch jemand RSA?

Patente

Viele Methoden, effizient ECC zu verwenden, waren oder sind patentgeschützt

z.B. Punktkompression, U.S. Patent 6,141,420, abgelaufen im Jahr 2014

Sorge um Hintertüren

Es besteht die Sorge, dass die Verwendung einer von einem Angreifer gewählten Kurve die Verschlüsselung unsicher machen kann.

Daher wurde D. Bernsteins Curve25519 etwa für OpenSSL anstelle von NIST-Kurven verwendet

3.6 Echtheit von Nachrichten

Wichtiger als die Vertraulichkeit von Information ist oft ihre Authentizität im Sinne von

(1) Authentizität des Senders oder Urhebers

(2) Unverfälschtheit der Nachricht

Zur Fehlererkennung bei Speicherung oder Übertragung von Daten werden ja bereits Prüfsummen (CRC-Codes) verwendet.

Die Idee besteht darin, einem Dokument eine Prüfsumme als eine Art Fingerabdruck mitzugeben, die es eindeutig identifiziert.

Techniken:

- Verschlüsselung
- Kryptographische Prüfsummen
- Hash-Funktionen
- Digitale Signatur

3.6.1 Hash-Funktionen

Soll die Unverfälschtheit der Nachricht gesichert werden, so kann man Hash-Funktionen verwenden.

Eine Hash-Funktion H bildet eine (variabel lange) Nachricht M auf einen Hash-Wert H(M) konstanter Länge ab.

Der Hash-Wert wird der Nachricht angehängt und bildet eine Art Fingerabdruck der Nachricht.

Hash-Funktionen haben die folgenden Eigenschaften:

- H ist nicht invertierbar, d.h. bei gegebenem Hash-Wert H(M) kann M nicht ermittelt werden.
- H(M) hängt von jedem Bit von M ab. Jede Modifikation von M muss zu einem anderen Hash-Wert führen.
- Es ist praktisch nicht möglich, zu einer gegebenen Nachricht M eine Nachricht M' zu finden, für die gilt : H(M) = H(M'). (**Schwache** Hash-Funktion)
- Ist es zusätzlich praktisch nicht möglich, ein Nachrichtenpaar M und M' zu finden, für das gilt : H(M) = H(M') so heißt die Hash-Funktion **stark**.

Hash-Funktion vs. Hash-Funktion

Kryptographische Hash-Funktionen

- Fester Wertebereich
- Geschwindigkeit nicht so wichtig, teilweise extra langsam (Passwort Hashes)
- Kollisionen vermeiden (Sicherheitsproblem)

Hash-Funktionen für Hash-Tabellen

- Wertebereich gleich Tabellengröße
- Schnell zu berechnen
- Kollisionen vermeiden (Performanceproblem)
- Kann vorhersagbar sein und Teile der Daten übernehmen (z.B. Modulo)

Konkrete Verfahren

Secure Hash Algorithm 2 (SHA-224, SHA-256, SHA-384, SHA-512)

- Standard seit 2001
- Sequentielle Verarbeitung von 512/1024-Bit-Blöcken,
- Erzeugt 224/256/384/512 bit langen Hashwert

Secure Hash Algorithm 1 (SHA-1)

- Anerkannter Standard (NIST) seit 1993
- Sequentielle Verarbeitung von 512-Bit-Blöcken,
- Aufteilung der Blöcke in 16 Worte a 32 Bit
- 80 Verarbeitungsschritte pro Block
- erzeugt 160 bit langen Hashwert

MD-5

- Vorgeschlagen von R. Rivest
- Sequentielle Verarbeitung von 512-Bit-Blöcken,
- Aufteilung der Blöcke in 16 Worte a 32 Bit
- 64 Verarbeitungsschritte pro Block
- erzeugt 128 bit langen Hashwert
- Kann gebrochen werden

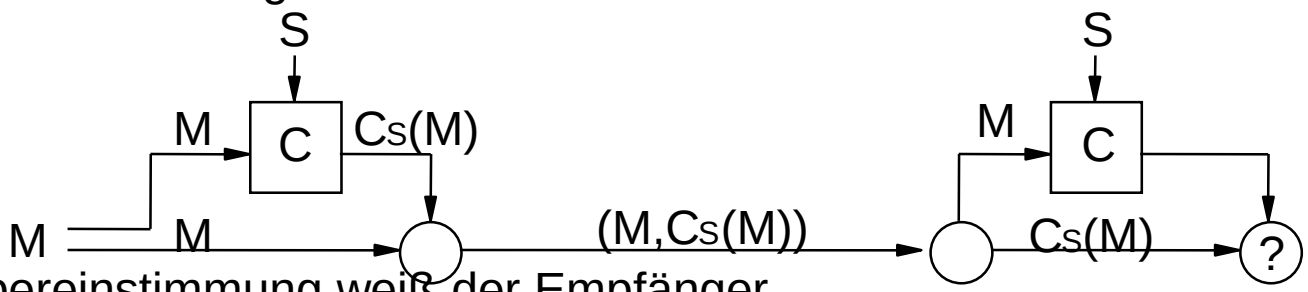
3.6.2 Kryptographische Prüfsummen

Ein Hashwert schützt nicht davor, dass jemand die Nachricht abfängt, ändert und mit einem neuen Hashwert versieht.

Um das zu verhindern, wird die Prüfsumme mit einem geheimen Schlüssel S kodiert (d.h. signiert) und der Nachricht hinzugefügt wird (MAC = Message Authentication Code):

$MAC = C_S(M)$

Der Empfänger trennt Nachricht und Prüfsumme und berechnet ebenfalls aus Nachricht M und Schlüssel S die Prüfsumme, die er mit der empfangenen Prüfsumme vergleicht.



Bei Übereinstimmung weiß der Empfänger,

- dass der Sender echt ist, da nur mit dem Schlüssel S dieselbe Prüfsumme herauskommt,
- dass die Nachricht nicht modifiziert wurde, sonst würde ebenfalls eine andere Prüfsumme resultieren.

3.6.3 Digitale Unterschrift

Message Authentication Codes sind schon relativ nah an einer „digitalen Unterschrift“.

Digitale Unterschriften sollen juristisch gleichwertig sein zu konventionellen Unterschriften:

- Identifikation
  - Auskunft über die Person des Unterzeichners
- Echtheit
  - Die Unterschrift bezeugt die Anerkennung des Dokuments.
- Abschluss
  - Die Unterschrift erklärt den Inhalt für richtig.
- Warnung
  - Der Unterzeichnende wird auf die juristische Bedeutung aufmerksam gemacht.

Anforderungen an DS-Verfahren

Zweifelsfreie Identität

- Die Unterschrift muss die Person eindeutig identifizieren.

Keine Wiederverwendbarkeit

- Die Unterschrift darf nicht von dem Dokument gelöst und anderweitig verwendet werden können.

Unveränderbarkeit

- Wenn die Unterschrift geleistet wurde, darf das Dokument nicht mehr verändert werden können.

Verbindlichkeit

- Es darf nicht abstreitbar sein, dass die Unterschrift geleistet wurde.

Signatur durch asymmetrische Verschlüsselung

Wie wir gesehen haben, wird bei asymmetrischer Verschlüsselung (RSA) mit dem öffentlichen Schlüssel des Empfängers codiert.

Dies garantiert die Vertraulichkeit der Nachricht, aber weder ihre Echtheit noch die des Senders.

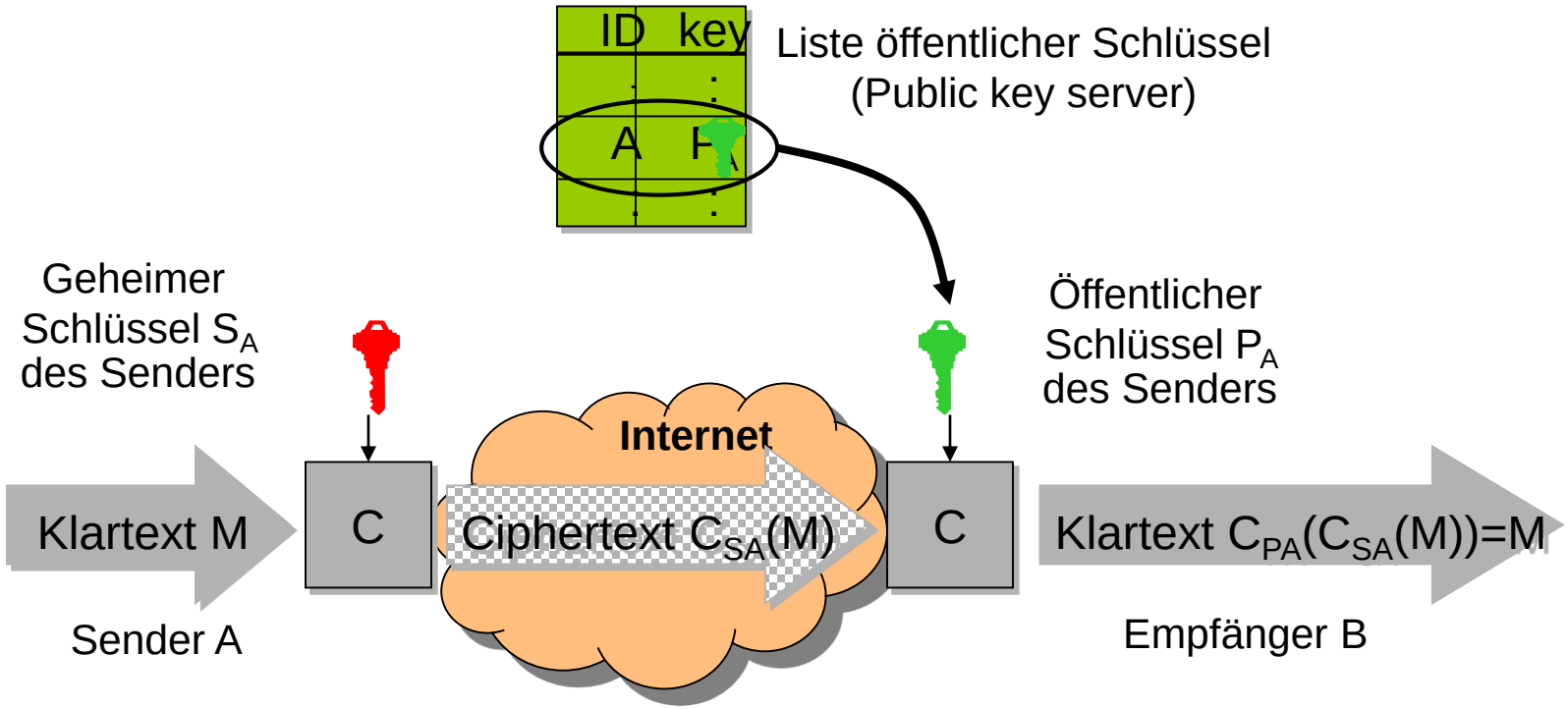
Um mit RSA die Authentizität zu gewährleisten, geht man umgekehrt vor und verschlüsselt mit dem geheimen Schlüssel des Senders (*Digitale Unterschrift*).

Dies sichert die Echtheit des Senders (kein anderer besitzt den geheimen Schlüssel), sichert aber nicht die Vertraulichkeit der Nachricht. (Jeder kann mit A's öffentlichem Schlüssel die Nachricht lesen.)

Daher kann man zusätzlich mit B's öffentlichem Schlüssel codieren, um sowohl Vertraulichkeit als auch Authentizität zu garantieren.

Digitale Unterschrift mit RSA

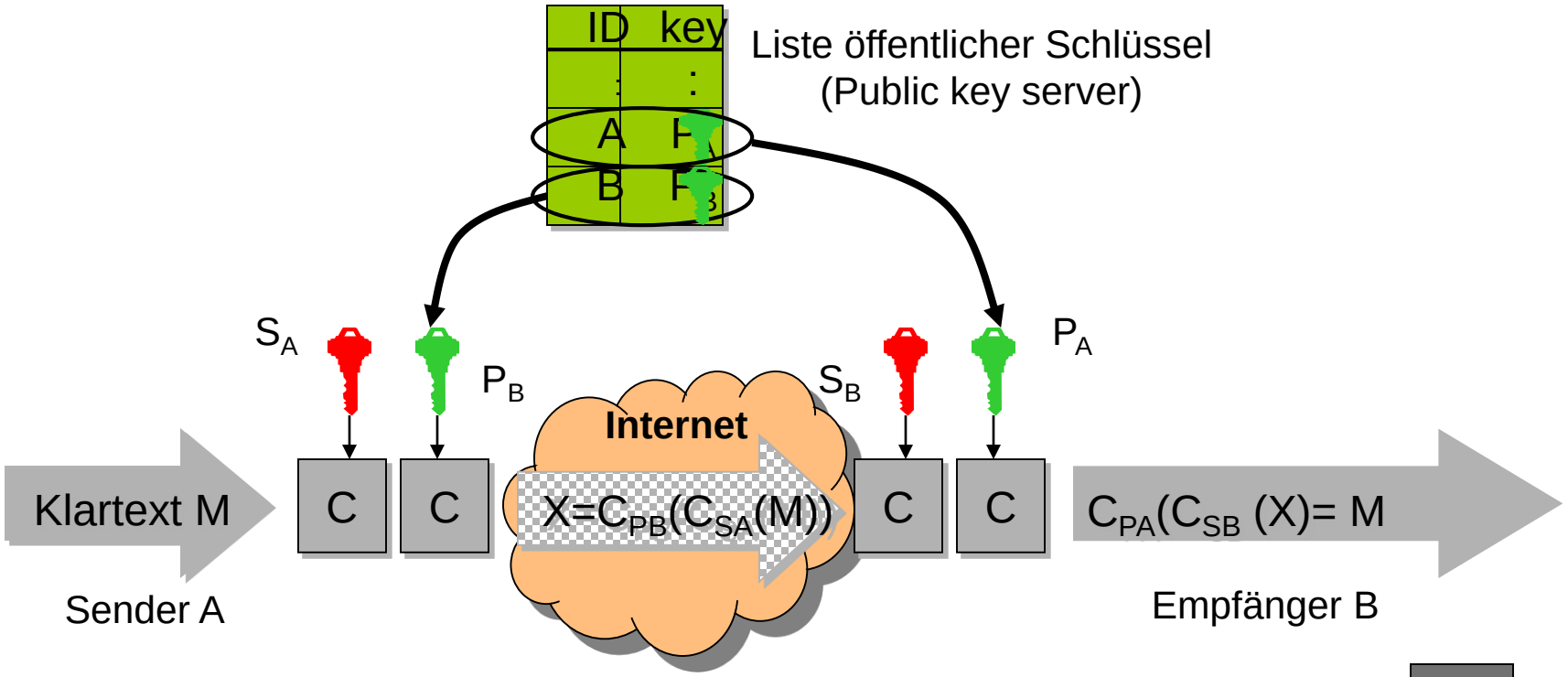
Da nur der Sender A selbst seinen geheimen Schlüssel kennt, kann eine Nachricht, die mit dem öffentlichen Schlüssel von A entschlüsselt werden kann, nur vom Sender A stammen.



Die Nachricht kann von jedem gelesen werden (keine Vertraulichkeit)

Will man zusätzlich zur Authentizität noch Vertraulichkeit, so kann man RSA zweimal anwenden:

Digitale Unterschrift + Vertraulichkeit



Digital Signature Standard

Der DSS wurde 1994 als Standard festgelegt (NIST).
Er verwendet asymmetrische Verschlüsselung.
Seine Sicherheit beruht auf der Schwierigkeit, den diskreten Logarithmus zu berechnen (ElGamal-Verfahren).
Bei der Unterschrift wird ein MAC mit SHA-1 erzeugt und in geeigneter Weise mit dem geheimen Schlüssel des Unterzeichners verknüpft.
Bei der Verifikation kann mit Hilfe des öffentlichen Schlüssels des Unterzeichners

- die Identität des Unterzeichners
- die Unverfälschtheit des Dokuments

überprüft werden.

Public-Key-Infrastruktur (PKI)

Bei Verwendung digitaler Unterschriften ist die Verfügbarkeit und die Echtheit (eindeutige Zuordnung zur Person) des öffentlichen Schlüssels essentiell.
Dazu benötigt man eine entsprechende Infrastruktur:
Aufbewahrung und Austausch öffentlicher Schlüssel (Key-Server)
Zertifizierung von Schlüsseln (Gehört der Schlüssel wirklich zu dieser Person?)
Umgang mit Zertifikaten

- Ausstellung
- Gültigkeitsdauer
- Rückruf
- Zertifizierung der Zertifizierungsstelle (Hierarchie)
- Gegenseitige Anerkennung (Cross-Certification)
- Transparente Integration in Standardwerkzeuge

Festgelegt in ISO-Standard X.509

Kritik an hierarchischer PKI

Sicherheit steht und fällt mit der Integrität der Zertifikatshierarchie
(z.B. bei S/MIME, HTTPS)

- Nutzer müssen vertrauenswürdige Wurzelzertifikate (CA) bestimmen
  - Auswahlproblem
  - meist wird die Liste des Herstellers ungefragt übernommen
- Längere Zertifikatsketten möglich
  - Z.B. Telekom → DFN → TUBIT → FG KBS
  - Kaum Kontrolle über (Sub-)Sub-CA
- Wurzelzertifikate können kompromittiert werden (z.B. DigiNotar)
- Trotzdem: Vor allem in abgeschlossenen Systemen einfacher zu realisieren (z.B. TU Portal, neue VBB-Monatskarte)

Technische Universität Berlin

Web of Trust

Idee: Zertifikate (bzw. deren Inhaber) sprechen sich gegenseitig das Vertrauen aus (z.B. PGP)

- Es entsteht ein Netzwerk aus Vertrauensbekundungen
- Wer das Vertrauen von vielen Teilnehmern hat (insbes. vertrauenswürdigen Teilnehmern), ist wahrscheinlich vertrauenswürdig
- Vorteile
  - Keine zentrale Instanz, die entscheidet bzw. ausfallen kann
  - Jeder kann selbst definieren, was „vertrauenswürdig“ ist
- Nachteile
  - Für Neueinsteiger schwer schnell genügend Unterschriften zu bekommen
  - Mit Fake-Zertifikaten kann man eine „Vertrauensblase“ erzeugen
  - Teilweise keine autorative Instanz für aktuellste Zertifikatversion

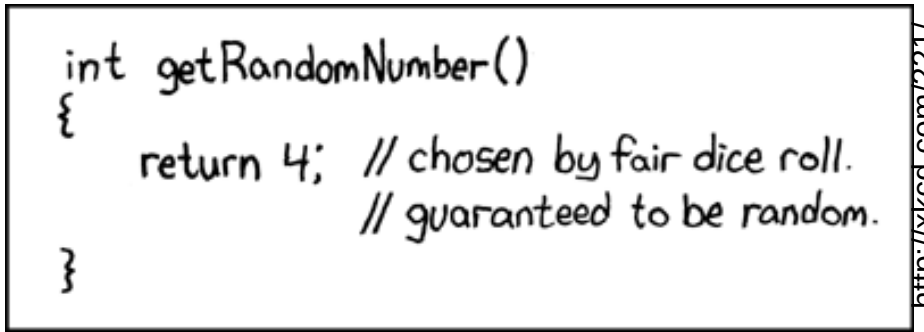
Starke Zufallszahlen

Viele kryptografische Verfahren und Protokolle benötigen Zufallszahlen

- Für Schlüsselgenerierung, IV, Nonce, Padding, Salt
- Lässt sich der Sitzungsschlüssel erraten, hilft es nichts ihn stark verschlüsselt auszutauschen!

Pseudozufallszahlen (PRNG)

- Funktion $f(s_n)=<r, s_{n+1}>$ erzeugt Pseudozufallszahl r aus Zustand s_n
- Startwert/Seed s_0 wird vor der ersten Zahl benötigt
- Deterministisch (gleiches s erzeugt gleiches r!)
- BSI Anforderungen
  - K1 – seltene Wiederholung von Teilen
  - K2 – verwechselbar mit echten Zufallszahlen
  - K3 – das nächste Bit trotz Kenntnis aller vorherigen nicht vorhersagbar
  - K4 – das nächste Bit trotz Kenntnis der Interna nicht vorhersagbar



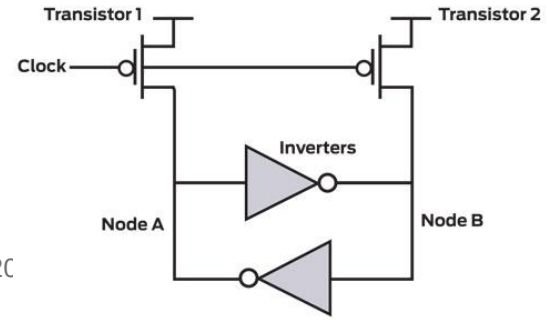
http://xkcd.com/221/

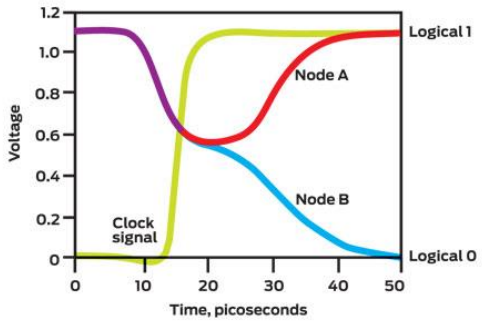
Echte Zufallszahlen

Echte Zufallszahlen lassen sich durch Beobachten von natürlichen Vorgängen bestimmen.

- Beispiele: Atmosphärisches Rauschen, Zerfallsprozesse, elektr. Wiederstand bei „rauschender“ Temperatur
- Quanten-Phänomene vs. Rauschen vs. Chaotische Systeme
- Entweder direkt benutzt oder als Seed für PRNG
- Beispiele:
  CPUs (Intel ab Ivy Bridge, VIA), Chipsätze, TPM-Modul, random.org

Beispiel: Schaltung mit zufälligem Wert





http://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator

Implementierungsfehler (Beispiele)

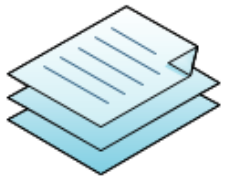
Zufall falsch initialisiert:

- PHP – Seed mußte früher per Hand initialisiert werden
  - mt_srand(time()); - Zeit in Sekunden vorhersagbar
  - mt_srand((double) microtime() * 100000); - Nur 16 Bit Zufall
- Debian OpenSSL-Patch (2008, CVE-2008-0166)
  - Valgrind zeigte einen Fehler im OpenSSL-Code, Debian-Maintainer entfernte Aufruf an zu vielen Stellen
  - Entropy-Pool für PRNG wurde nicht gefüllt → vorhersagbare Schlüssel und Zertifikate (sehr viele Server betroffen)

Kryptographie falsch Implementiert

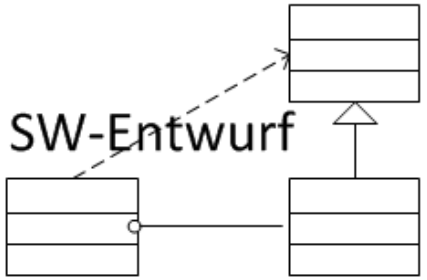
- Frühere EC-Karten – Übersetzung von Hexadezimal nach Dezimal
- ATEN KN9116 IP KVM (CVE-2009-1473)

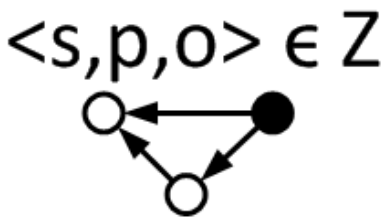
Sicherheits-
konzept



Risiko

Bedrohungen





Security | **Safety** | **Schutz** | **Sicherheit**

Weitere Informationen

Bauer, F.L.: *Decrypted Secrets*, Springer-Verlag, 2000

Stinson, D.R.: *Cryptography – Theory and Practice*, CRC Press, 2000, Chapter 1-4

Schneier, B.: *Applied Cryptography*, John Wiley, 1996 (deutsche Ausgabe bei Person Studium, 2006)

Stallings,W.: *Cryptography and Network Security*, 4th ed. Prentice Hall, 2006, Chapter 2-13

AES Rijndael http://csrc.nist.gov/encryption/aes/rijndael

Cormen,T. et al.: *Introduction to Algorithms*, MIT Press, 1990 (RSA Korrektheit)

Schmeh,K.: *Kryptografie und Public-Key-Infrastrukturen im Internet,* dpunkt-Verlag, 2001

Garfinkel, S.; Spafford, G.: *Practical Unix and Internet Security*, 3rd ed. O'Reilly, 2003, Chapters 3, 8, 19

Beutelsbacher, A. et al.: *Moderne Verfahren der Kryptographie*, 5. Aufl., Vieweg, 2004

Knuth,D.E.: *The Art of Computer Programming*, I-III, Addison-Wesley, 1997

Wagner, N.: The Laws of Cryptography with Java Code. http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf