IT-Sicherheit

Kapitel 2: Verschlüsselung (Teil 1)

- Symmetrische Verschlüsselung
- Base64 Codierung
- Asymmetrische Verschlüsselung
- Praktische Aspekte bei der Verschlüsselung
- Zufallszahlen
- Schlüsselmanagement

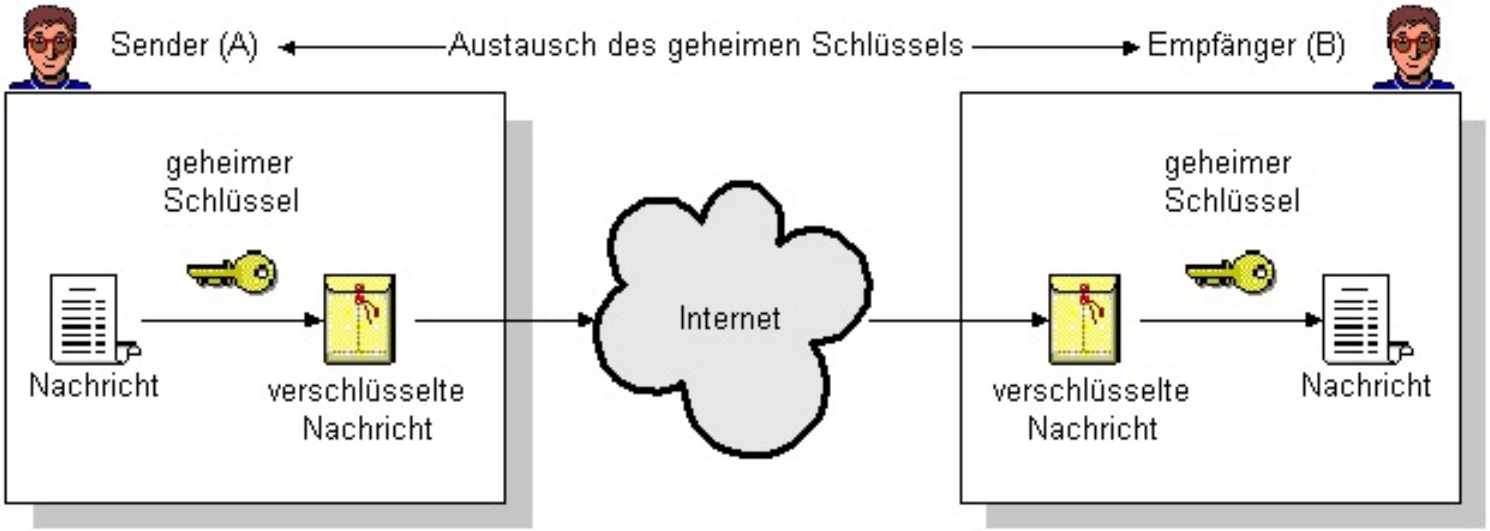
Prof. Dr. Reiner Hüttl, FH Rosenheim, © 2021, 15.03.21

Worum geht es?

- Welche Arten der Verschlüsselung gibt es
- Was sind die wichtigsten Algorithmen und Verfahren
- Wie wende ich Verschlüsselung praktisch an
- Auf was muss ich alles achten damit die Verschlüsselung wirklich mehr Sicherheit bringt

Symmetrische Verschlüsselung

- Ein zentraler Schlüssel für Ver- und Entschlüsselung
- Nachteil: Austausch des geheimen Schlüssel erforderlich
- Vorteile:
  - sehr schnell,
  - in HW implementierbar



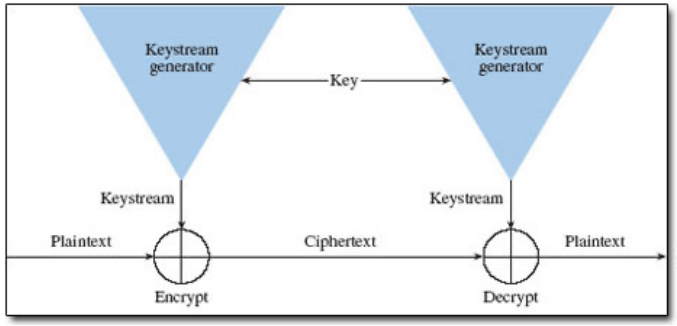
Varianten von symmetrischen Chiffren

- **Stromchiffren:**
  - Nehmen einen Key fester Länge
  - Generieren aus dem Key einen Strom beliebiger Länge, bestehend aus Pseudozufallszahlen
  - Konvertieren jeweils ein Bit des Klartextes zu Chiffretext mittels XOR
  - Beispiele: RC4 (UNSICHER seit 2013!!!), ChaCha20
- **Blockchiffren**
  - Bearbeiten Klartext und Chiffretext in Blöcken (z.B. 64 Bit, 128 Bit)
  - Haben einen Modus (block cipher mode)
  - Beispiele: DES, DES3 (UNSICHER!!!), AES, Twofish, Serpent
- **Padding**
  - Auffüllen des letzten unvollständigen Blocks mit „regelmäßigen“ Mustern
  - Einige Algorithmen (meistens Blockkchiffre) benötigen volle Blöcke
  - Beispiele: PKCS5 Padding, W3C Padding, ISO-Padding, ESP-Padding

unregelmäßig aber rekonstruierbar

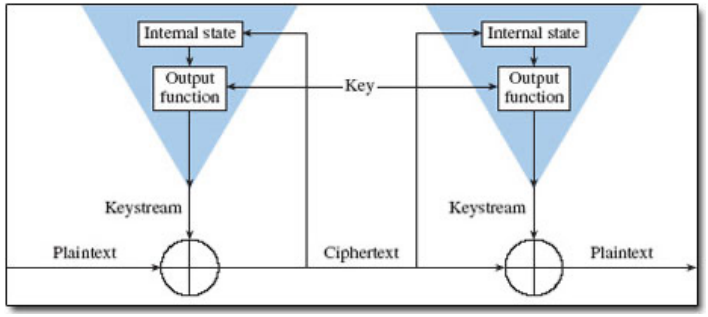
Varianten von Stromchiffren

- Synchrone Stromchiffrierung
  - generiert den Schlüsselstrom unabhängig vom Klar- oder Schlüsseltext



From Schneier, *1996, Figure 9.6*

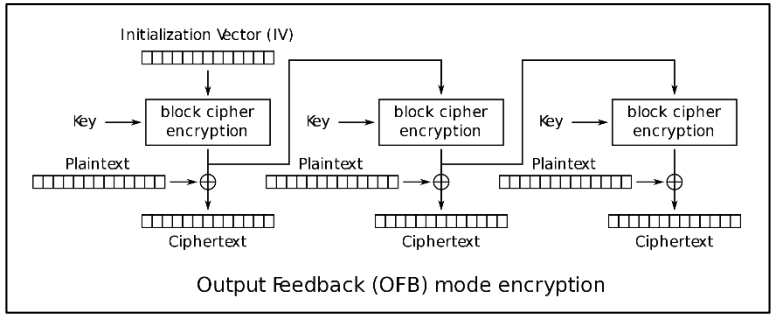
- Selbstsynchronisierende Stromchiffrierung
  - Schlüsselstrom hängt von vorhergehenden verschlüsselten Bits ab

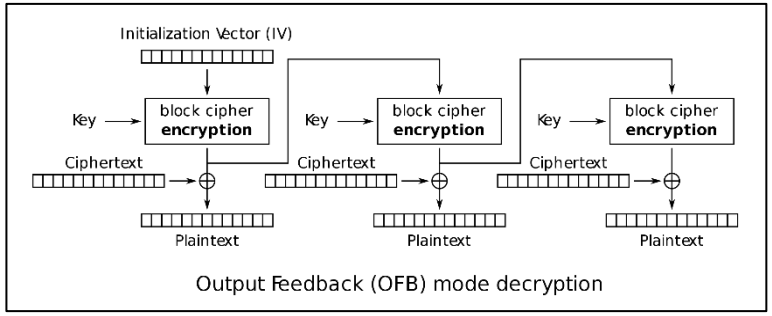


From Schneier, *1996, Figure 9.8*

Blockchiffren können als Stromchiffren implementiert werden

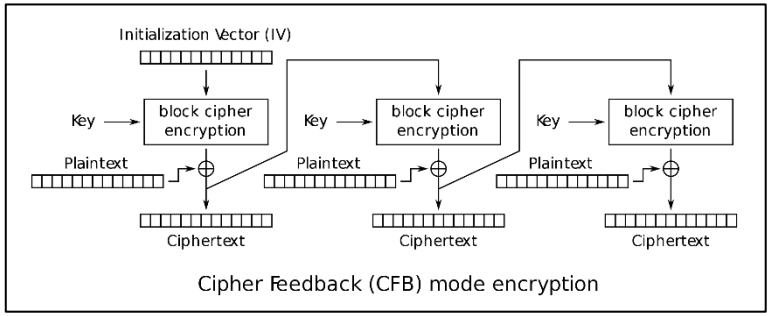
- **OFB, Output Feedback Mode**
- Ist eine synchrone Stromchiffrierung
- Der Schlüsselstrom kann vorausberechnet werden

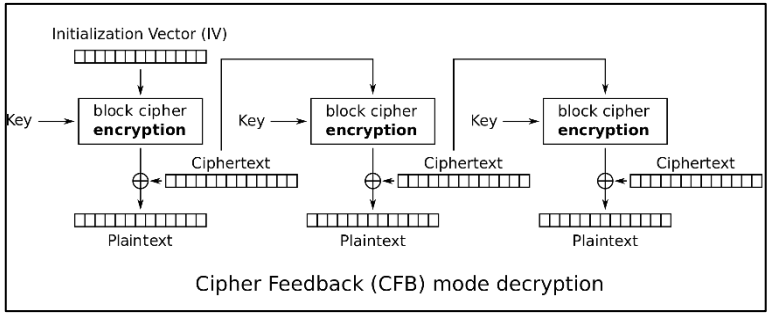




https://de.wikipedia.org/wiki/Output_Feedback_Mode

- **CFB Cipher Feedback Mode**
- Selbstsynchronisierende Stromchiffrierung
- Fehler im IV oder Ciphertext wirken sich nur auf zwei weitere Blöcke aus





https://de.wikipedia.org/wiki/Cipher_Feedback_Mode

Das Standard Verschlüsselungsverfahren Advanced Encryption Standard (AES)

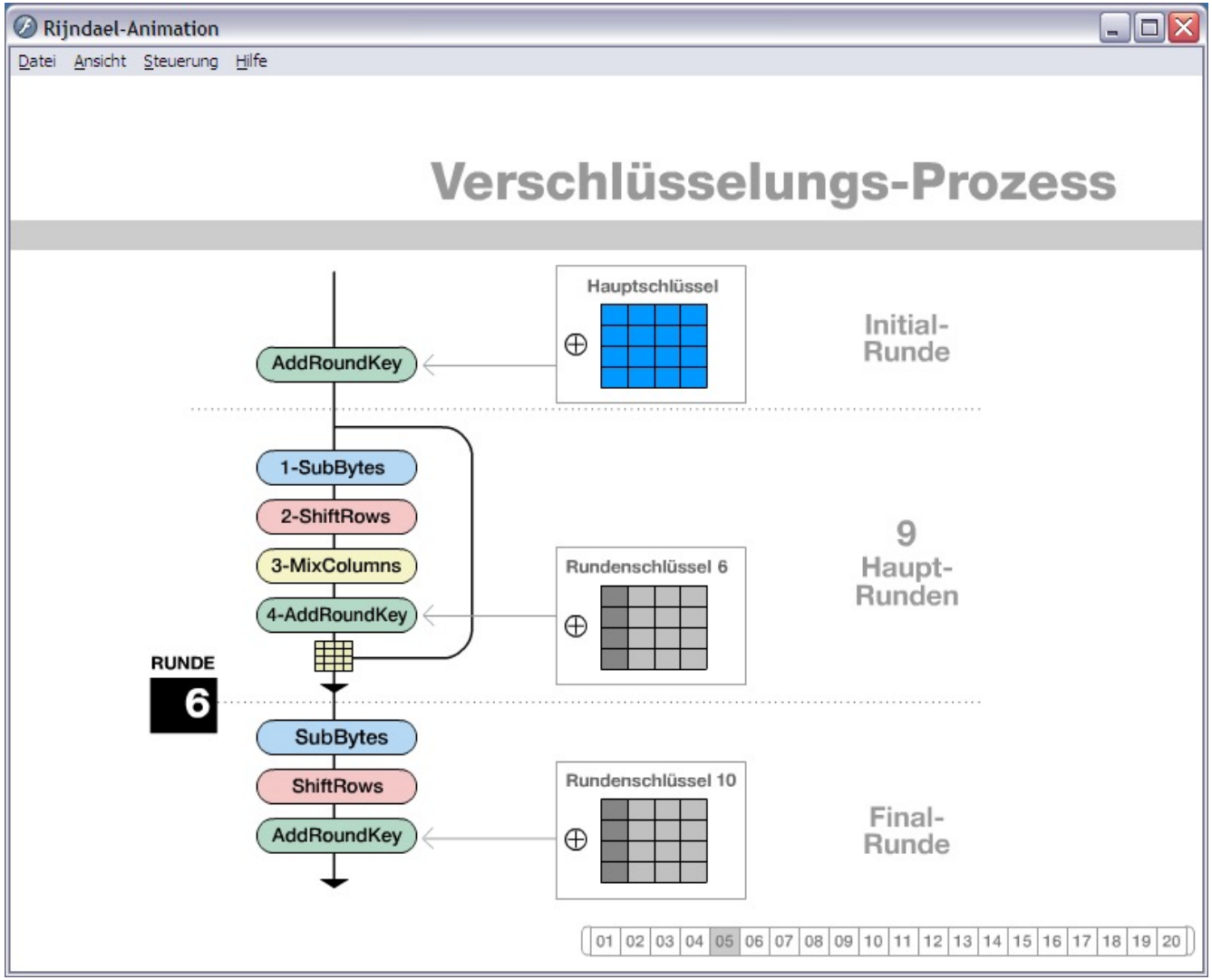
- Anforderungen an AES bei Wettbewerb von NIST 2001
  - Blockchiffre mit 128, 192 und 256 Bit
  - Mathematische Rechtfertigung der Sicherheit
  - Einfachheit des Designs
  - Flexibilität bezüglich Blockgrößen und Schlüssellängen
  - Effizienz
  - HW- und SW-Implementierung einfach

Alter Standard, heute unsicher, nicht mehr einsetzen!

- Offizieller Nachfolger von DES, basiert auf **Rijndael Algorithmus**
  - In einem offenen Verfahren aus 15 Vorschlägen ausgewählt
  - Alle Design-Kriterien vollständig veröffentlicht
  - Verschlüsselt Blöcke mit fester Länge von 128 Bit (16 Byte)
  - Schlüssellänge wahlweise 128, 192 oder 256 Bits

Empfohlene Schlüssellänge

Der Algorithmus von AES



Quelle: Cryptool Portal,
http://www.cryptool.org/de/
Cryptool 1, AES Animation

„Einzelverfahren“ \„Visualisierung von Algorithmen“ \„AES“ \„Rijndael-Animation“

Kleine Übung: Simulation von AES anschauen

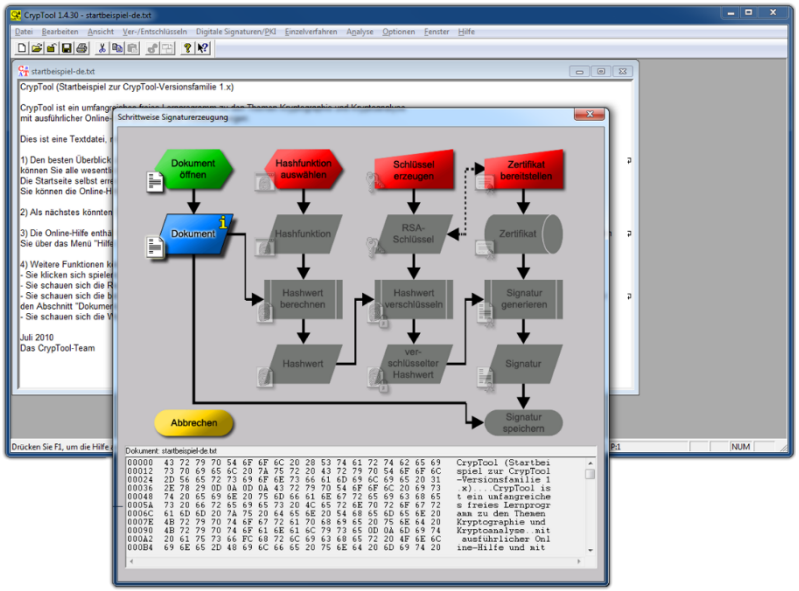
- **CrypTool 1** herunterladen und installieren (ACHTUNG nicht CrypTool 2)
  https://www.cryptool.org/de/cryptool1

- Das Programm **CrypTool 1 (CT1)** ist ein kostenloses Windows-Programm für Kryptographie und Kryptoanalyse. Diese E-Learning-Software gibt es in 6 Sprachen und sie ist die weltweit verbreiteste ihrer Art.

- Prof. Bernhard Esslinger, Universität Siegen

- Gehen sie in das Menu
  „Einzelverfahren“
  „Visualisierung von Algorithmen“
  „AES“
  „Rijndael-Animation“

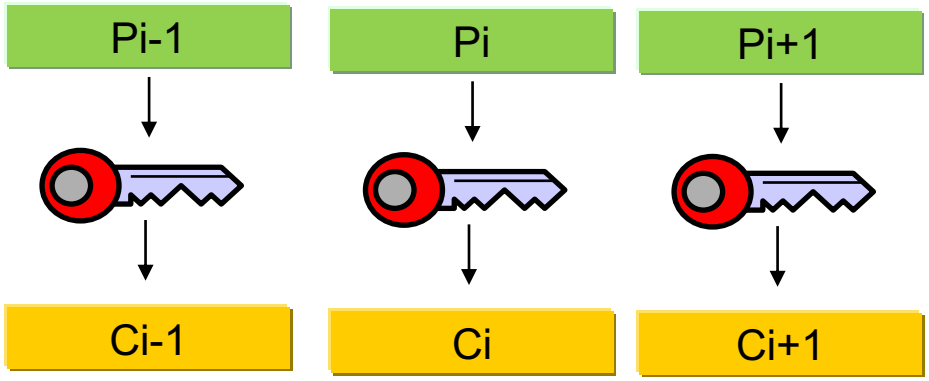
- Gehen sie die Animation schrittweise durch



Betriebsmodus für Blockchiffre: ECB

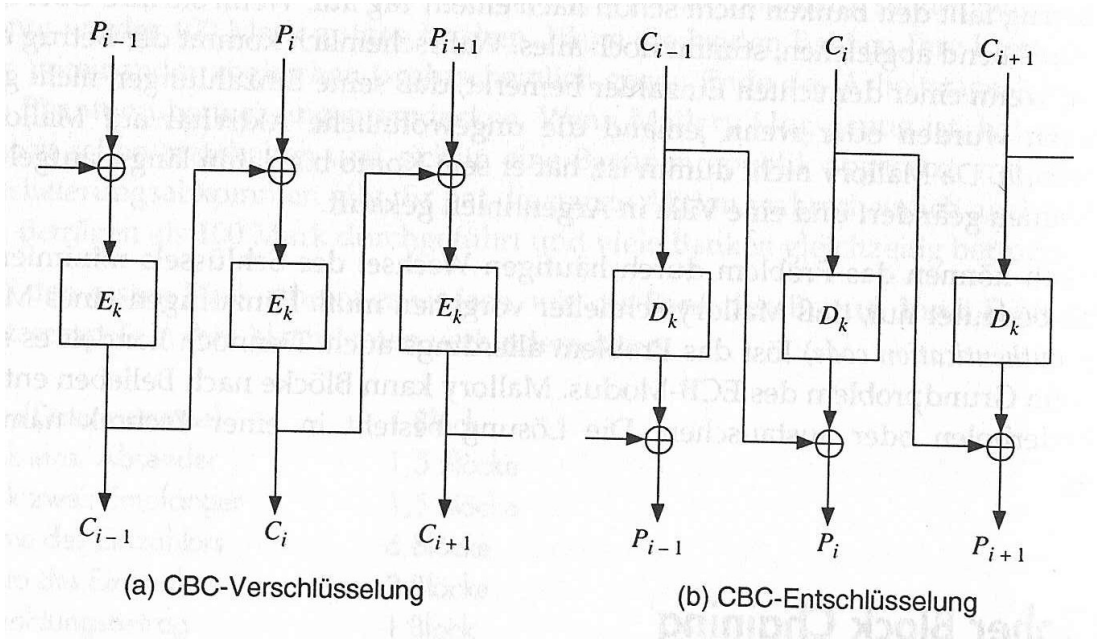
- Ein symmetrisches Verfahren (z.B. AES) kann in verschiedenen Modi betrieben werden

- **Electronic Code Book (ECB)**
  - Ein Klartextblock wird in einen Chiffretextblock verschlüsselt
  - Gleiche Klartextblöcke erzeugen gleiche Chiffretextblöcke
  - Alle Blöcke können unabhängig voneinander chiffriert werden
  - Einfacher für Kryptoanalytiker zu entschlüsseln (stereotype Anfänge und Enden)
  - Einfache Bitfehler haben keinen Einfluss auf andere Blöcke
  - Bei verlorenen Bits ist ein Neusynchronisation der Blockgrenzen erforderlich



Betriebsmodus für Blockchiffre: CBC

- **Cipher Block Chaining (CBC)**
  - Rückkopplung: Verschlüsselung eines Blocks hängt Vorgängerblöcken ab
  - Klartext wird vor Verschlüsselung mit vorherigen Chiffretextblock mit XOR verknüpft
  - **Initialisierungsvektor** für ersten Block notwendig:
    nicht geheim, eindeutig rekonstruierbar, für jede Nachricht anders
  - Problem: Fehlerfortpflanzung
    Bitfehler: 1-Bitfehler in Ciphertext führt zu Fehler in Block und Folgeblock
    Synchronisationsfehler: keine Erholung mehr



(a) CBC-Verschlüsselung (b) CBC-Entschlüsselung

$$C_i = E_K(P_i \oplus C_{i-1})$$
$$P_i = C_{i-1} \oplus D_K(C_i)$$

Quelle: Bruce Schneier: Angewandte Kryptographie, Addison-Wessley, 2005

Achtung bei Wahl des Initialisierungsvektors

- Initialisierungsvektor
  - Muss nicht geheim gehalten werden (einfach mit Ciphertext mitschicken)
  - sollte zufällig sein (z.B. SecureRandom)
  - Sollte für jede Nachricht anders sein
  - Ist genauso groß wie die Blocksize, bei AES immer 128 Bit
- Populäre Fehler
  - Key als IV verwenden
    - Damit kann im dümmsten Fall der Key gelesen werden [1]
  - IV auf 0 oder anderen festen Wert setzen
    - IV bringt den Zufall ins System - deswegen zufällig würfeln!
  - IV aus Passwort generieren
    - Lieber IV zufällig würfeln!

[1] https://crypto.stackexchange.com/questions/16161/problems-with-using-aes-key-as-iv-in-cbc-mode

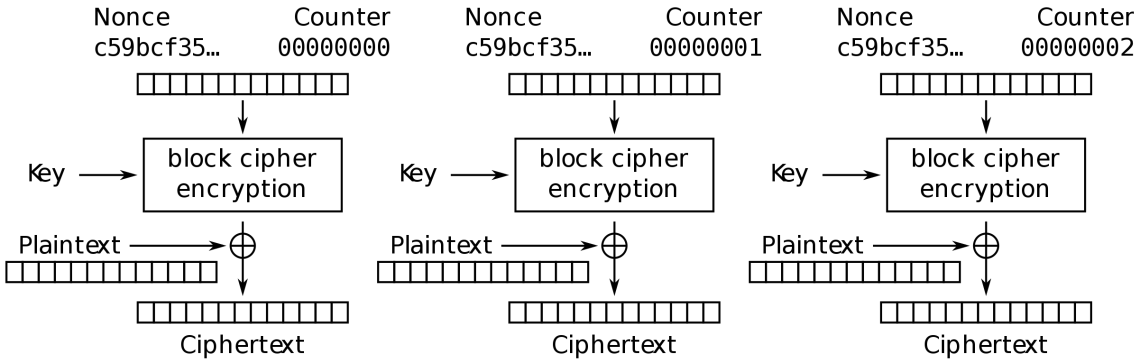
Betriebsmodus CTR Counter Block Mode

- Für jeden Block wird ein neuer unberechenbarer keystream block berechnet:
  Keystream block = IV(Nonce) + current counter + encryption key

Nonce = Number used only once

- Vorteile:
  - Schlüssel hängt nicht vom Schlüssel für den vorherigen Block ab
  - Ver- und Entschlüsselung kann parallel durchgeführt werden
  - Schlüsselstrom kann vorausberechnet werden (bei Betrieb als Stromchiffre)
  - Bitfehler betreffen nur einen Block

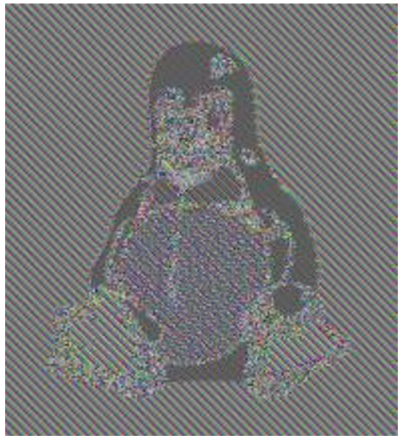
$$C_i = P_i \oplus E_K(ctr_i)$$



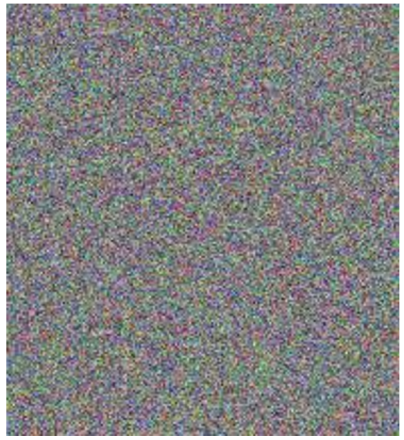
Counter (CTR) mode encryption

ECB vs. CBC

- Populärer Fehler bei Verschlüsselung: Falschen Modus verwenden
- **ECB ist schlecht**, besser CBC (oder GCM - kommt später)



ECB



CBC

Anwendung von Kryptographie ist schwierig

- Ich verwende AES/CBC, alles gut? **NEIN**
  - Angreifer kann Ciphertext ändern und damit Einfluss auf den Plaintext nehmen
  - Angreifer kann **Padding Oracle Attack** durchführen und Plaintext lesen
  - siehe https://www.arxumpathsecurity.de/blog/2019/10/16/cbc-mode-is-malleable-dont-trust-it-for-authentication
- Weitere populäre Fehler
  - Niemals CBC, CTR, etc. ohne Authentication verwenden!
    - Angreifer kann sonst Plaintext lesen / verändern
- Deswegen: Verschlüsselung mit Authentication absichern
  - MAC Message Authentication Code
    MAC = Hashfunktion die geheimen Schlüssel verwendet

Authentisierung
Ziel der sicheren Identifikation einer Person oder einer Maschine

MAC wird im Kapitel Digitale Signaturen genauer behandelt

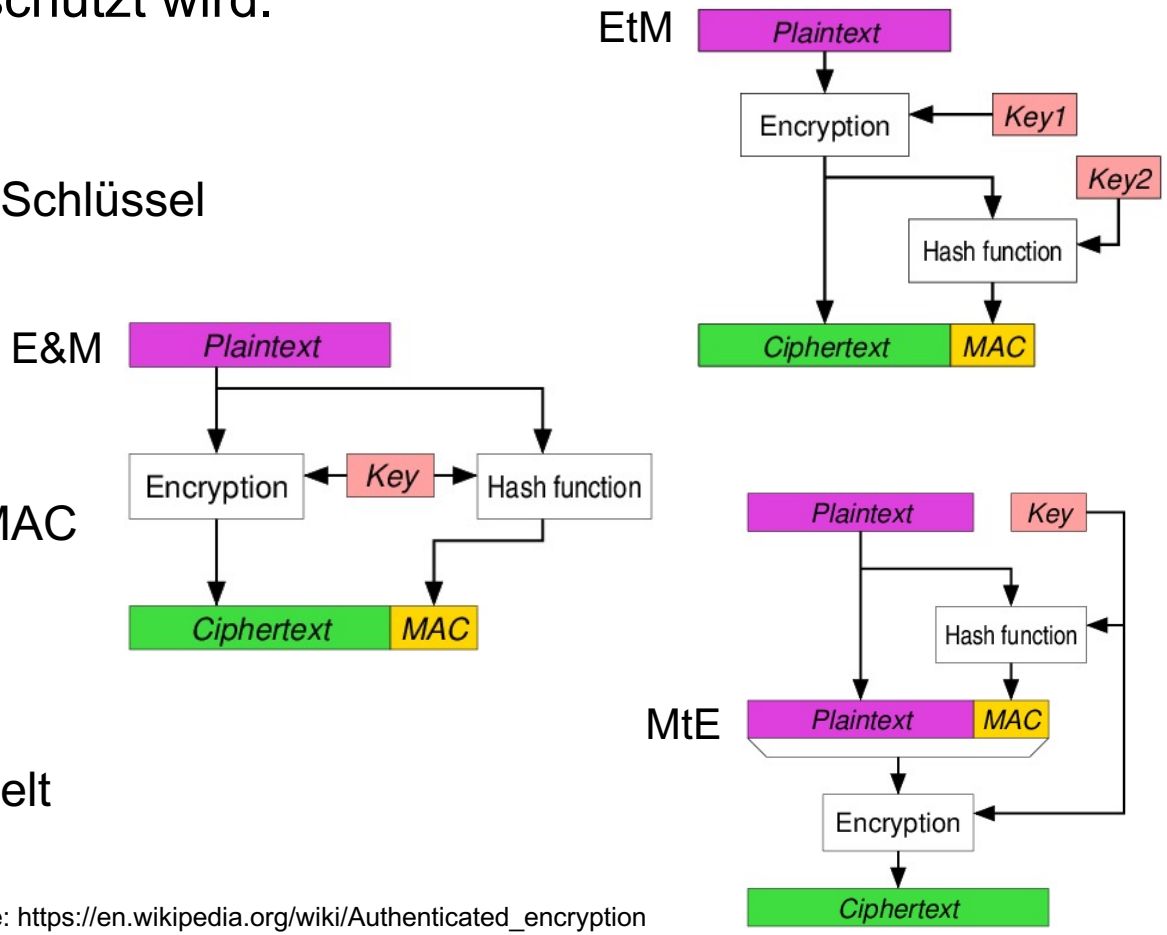
Authenticated Encryption

- **Authentisierte Verschlüsselung** Verschlüsselungsverfahren heißen authentisiert, wenn nicht nur die Vertraulichkeit, sondern auch die Integrität der zu verschlüsselnden Daten geschützt wird.

- Encrypt-then-MAC
  - **Hohe Sicherheit** wenn beide Schlüssel unterschiedlich

- Encrypt-and-MAC
  - MAC auf Plaintext
  - Plaintext verschlüsselt ohne MAC

- MAC-then-Encrypt
  - MAC auf Plaintext
  - MAC und Plaintext verschlüsselt



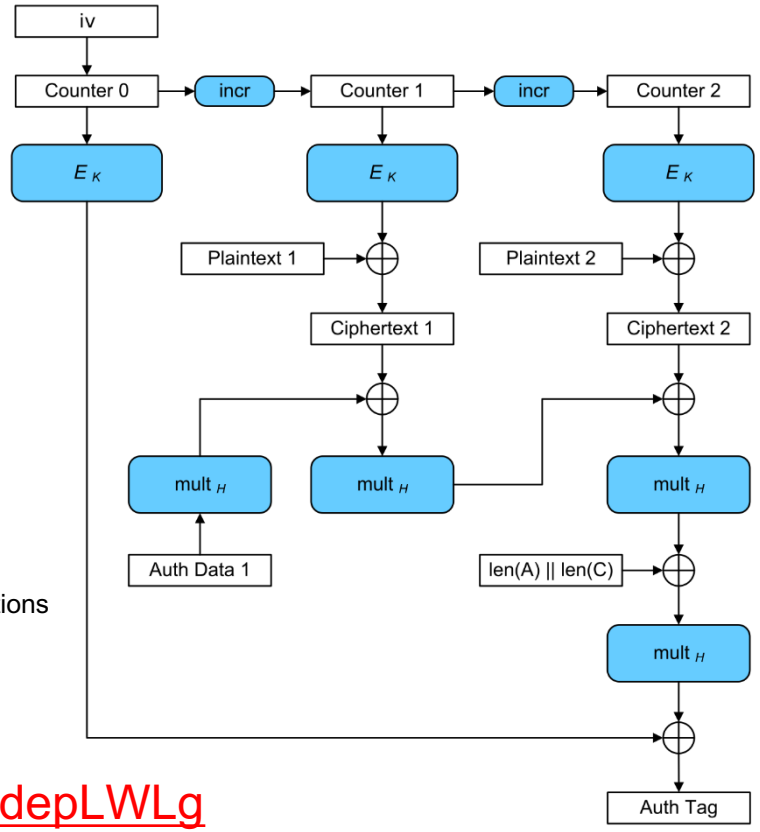
Quelle: https://en.wikipedia.org/wiki/Authenticated_encryption

Authenticated Encryption

- Encrypt-then-MAC
  - AES+CBC+HMAC-SHA256
    https://www.mkammerer.de/blog/encrypt-something-with-aes-how-hard-can-it-be/
  - ChaCha20 (encrypt) + Poly1305 (MAC)
    https://github.com/phxql/chacha20-poly1305-java
- Populäre Fehler
  - Gleichen Key für AES und HMAC verwenden
    - Ist in gewissen Fällen ok, wenn es anders geht: nicht machen
  - MAC-then-encrypt
    - AES(HMAC(Plaintext) || Plaintext)
    - Es gibt Padding Attacken (z.B. bei SSL)
  - MAC-and-encrypt
    - AES(Plaintext) || HMAC(Plaintext)
    - Keine Integrität auf Ciphertext

Betriebsmodus GCM Galois Counter Mode

- Authentifizierter Verschlüsselungsmodus mit assoziierten Daten (AEAD)
  - Schnell, da Parallelisierung möglich
  - Kann auch als Stand-alone MAC verwendet werden
  - Akzeptiert IV (Nonce) beliebiger Länge
  - Auth Tag enthält Auth Data, IV und Ciphertext



Ek Encryption with Key k

Len (a) = len (Auth Data)

Len (C) = len (ciphertext)

Mult H Galois field multiplications

- Video zur Veranschaulichung von GCM

https://www.youtube.com/watch?v=R2SodepLWLg

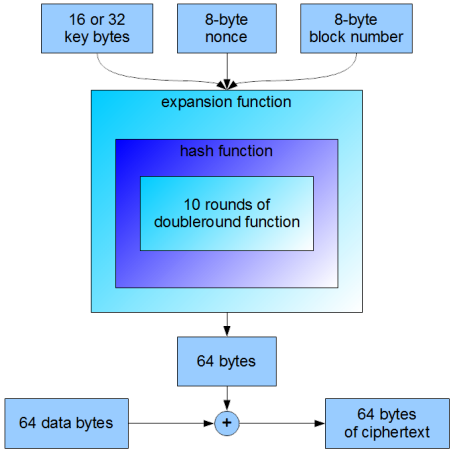
https://de.qwe.wiki/wiki/Galois/Counter_Mode

AEAD Authenticated Encryption with Associated Data

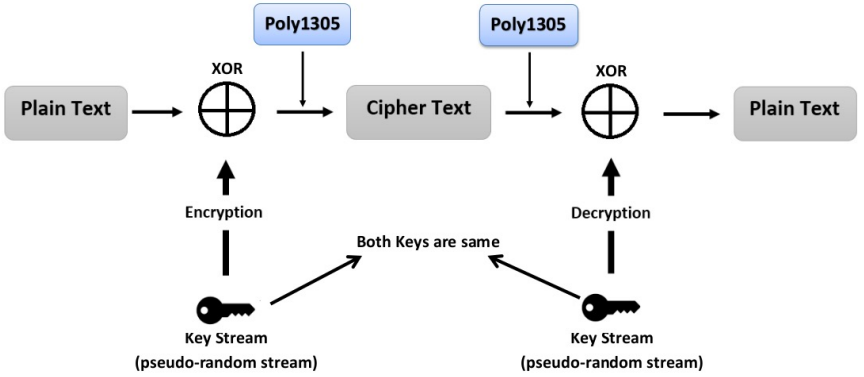
- AEAD kombiniert Vertraulichkeit, Authentizität und Integrität
  - Sind automatisch sicher gegen Veränderung des Ciphertexts
  - Kombinieren Verschlüsselung und MAC in ein integriertes Protokoll
  - z.B. AES-GCM
    - Verwendet eine 96 bit Nonce
    - Nonce (IV) muss nicht zufällig sein, darf aber bei gleichem Key **NIEMALS** wiederverwendet werden
  - Weitere Variante AES-EAX
- Video zur Veranschaulichung von AEAD
  https://www.youtube.com/watch?v=od44W45sCQ4

AEAD mit Stromchiffre ChaCha20 und Poly1305

- ChaCha20
  - Symmetrisches Stromchiffre basierend auf Salsa20 von Daniel Bernstein
  - Implementierung einfacher als AES-GCM
  - Einsatz in IPSec, TLS, OpenSSL, OpenSSH
- Poly1305 ist als MAC viel schneller als HMAC



Block Diagram of Salsa20 Algorithm
http://www.crypto-it.net/eng/symmetric/salsa20.html



ChaCha20 Poly1305 Encryption and Decryption scheme
https://javainterviewpoint.com/chacha20-poly1305-encryption-and-decryption/

Base64 Kodierung

Base64 ist eine Kodierung und keine Verschlüsselung!

- Zur Übertragung von 8-Bitgruppen über nicht 8-bitsichere Wege
- Darstellung von Binärdaten mit 64 druckbaren ASCII-Zeichen
- Prinzip:
  - 24 Bit in 4 Teile zu 6 Bit zerlegen
  - Jede 6 Bitfolge auf 8 Bit erweitern
- Nachteil: Nachricht wird um 33% länger
- Anwendung:
  - Schlüssel, Signaturen und Zertifikate werden häufig BASE64 kodiert abgespeichert oder übertragen

Beispiel für Base64 Kodierung

Base64 Alphabet

Wert	Zeichen	Wert	Zeichen	Wert	Zeichen	Wert	Zeichen
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	64	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

Beispiel abcde			
Quelle	**Binärdarstellung**	**Base64**	
a	01100001		
b	01100010		
c	01100011		
24-Bit	011000010110001001100011		
6-Bit	011000	24	Y
	010110	22	W
	001001	9	J
	100011	35	j
d	01100100		
e	01100101		
	____		
24-Bit	011001000110010100____		
6-Bit	011001	25	Z
	000110	6	G
	010100	20	U
	____		=
YWJjZGU=			