Studienwoche 2022

AES

05. Juni 2023

ubique

Studienwoche 2022

AES

1. Probleme von Substitutionschiffren
2. Block-Chiffren
3. Initial Vektor
4. AES
5. AES-ECB
6. AES-CBC
7. AES-GCM

AES

Probleme mit Substitutionschiffren

ubique

Probleme mit Substitutionschiffren

- Wegen Eindeutigkeit kann die Buchstabenverteilung nicht verändert werden

- Bei bekannter Satzstruktur, können Buchstaben erraten werden

- Gleicher Klartext hat gleichen Ciphertext

    - Falls Struktur von Text bekannt, kann die Chiffre leicht gebrochen werden
    - Falls z.B. ein Brief verschlüsselt wird
    - Briefe haben eine Begrüssungsformel (Sehr geehrte Damen und Herren)
    - Briefe haben eine Schlussformel (Mit freundlichen Grüssen)

Probleme mit Substitutionschiffren

- Sind anfällig auf «Chosen/Known Plaintext-Attacks»
- Bei «Chosen Plaintext Attacks» wählt der Angreifer den Plaintext und bekommt Ciphertext
- Beispiel Cäsar-Chiffre:
  - Angreifer: Verschlüssel mir «ich bin ein klartext»
  - Orakel: YSX RYD UYD ABQHJUNJ
  - → Die Buchstaben wurden um 16 verschoben

AES

Block Chiffren

ubique

Block Chiffren

- Block Chiffren arbeiten auf Blöcken bestimmter Länge
- Für jeden Block gibt es zwei Algorithmen:
  - Verschlüsselung: nimmt einen (Klartext)-Block und einen Schlüssel und generiert einen neuen Block
  - Entschlüsselung: nimmt einen (Ciphertext)-Block und einen Schlüssel und kehrt den Verschlüsselungsblock um

AES

Block Chiffren

- Um Nachrichten beliebiger Länge zu verschlüsseln muss die Block-Chiffre mehrmals auf verschiedene Blöcke angewandt werden

- Entsprechend gibt es verschieden «Betriebsmodi»

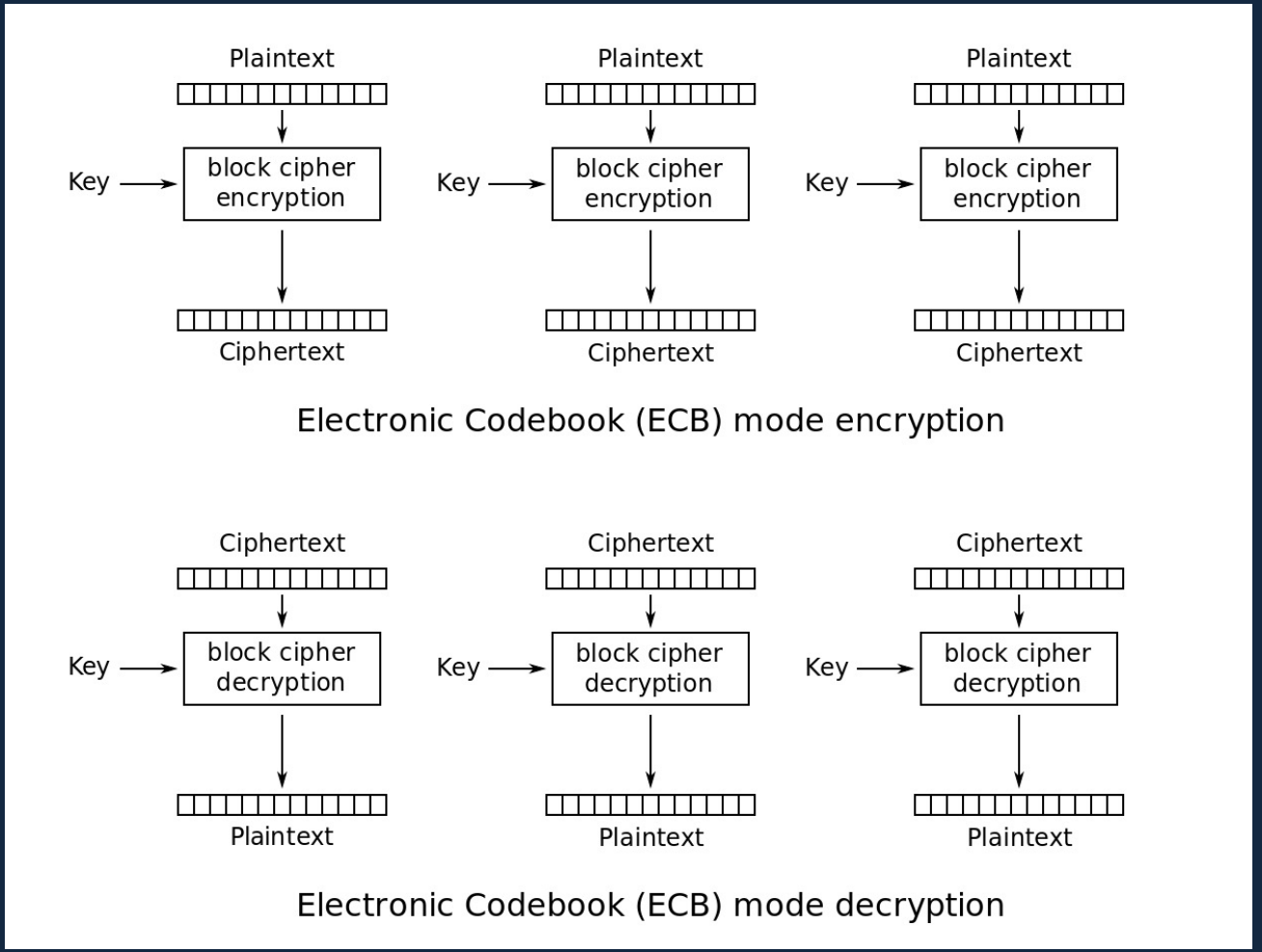
  - Electronic Code Book (ECB): jeder Block wird einzeln verschlüsselt

  - Cipher Block Chaining (CBC): jeder Block wird mit dem vorhergehenden Verknüpft (XOR). Es braucht einen Initialiserungsvektor

  - Cipher/Output Feedback Mode: Die Block-Chiffre wird nur zur Erstellung von pseudozufälligen Zahlen verwendet. Es braucht einen Initialisierungsvektor

  - Counter Mode: Hier wird eine Zufallszahl mit einem «Counter» als Input für die Block-Chiffre gewählt. Die so pseudozufälligen Zahlen werden mit dem Klartext XOR’d

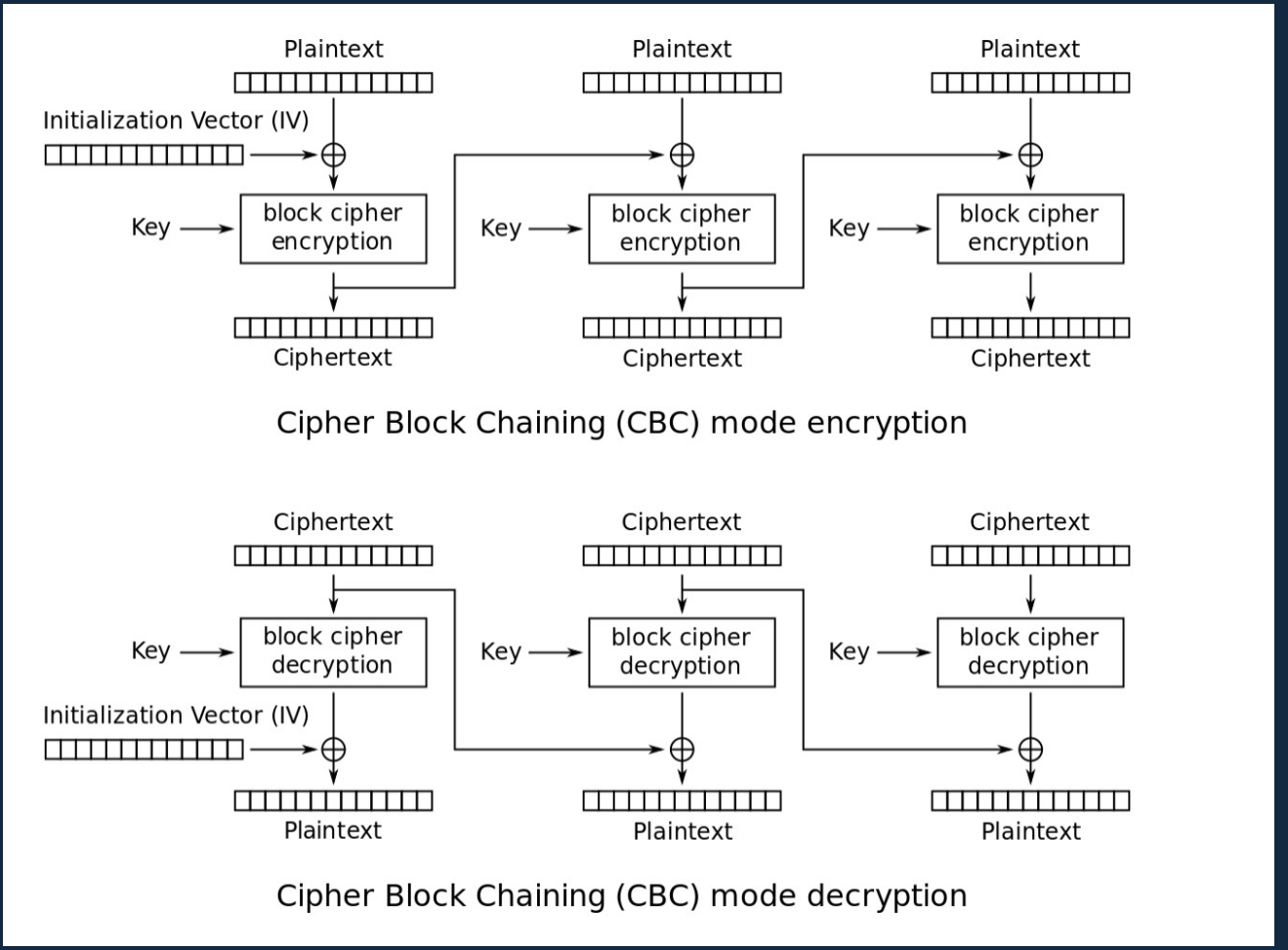
Electronic Code Book



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

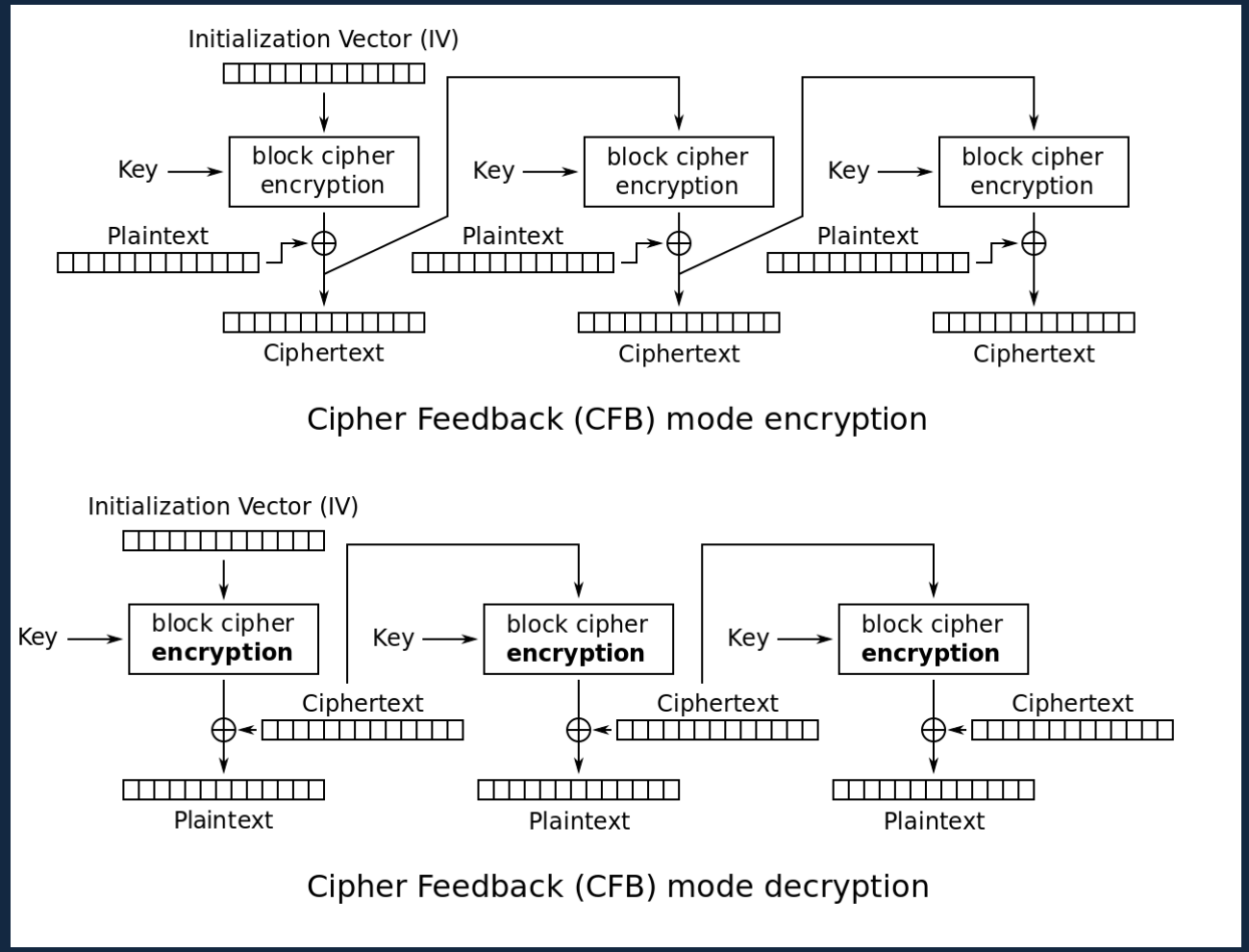
AES

Cipher Block Chaining



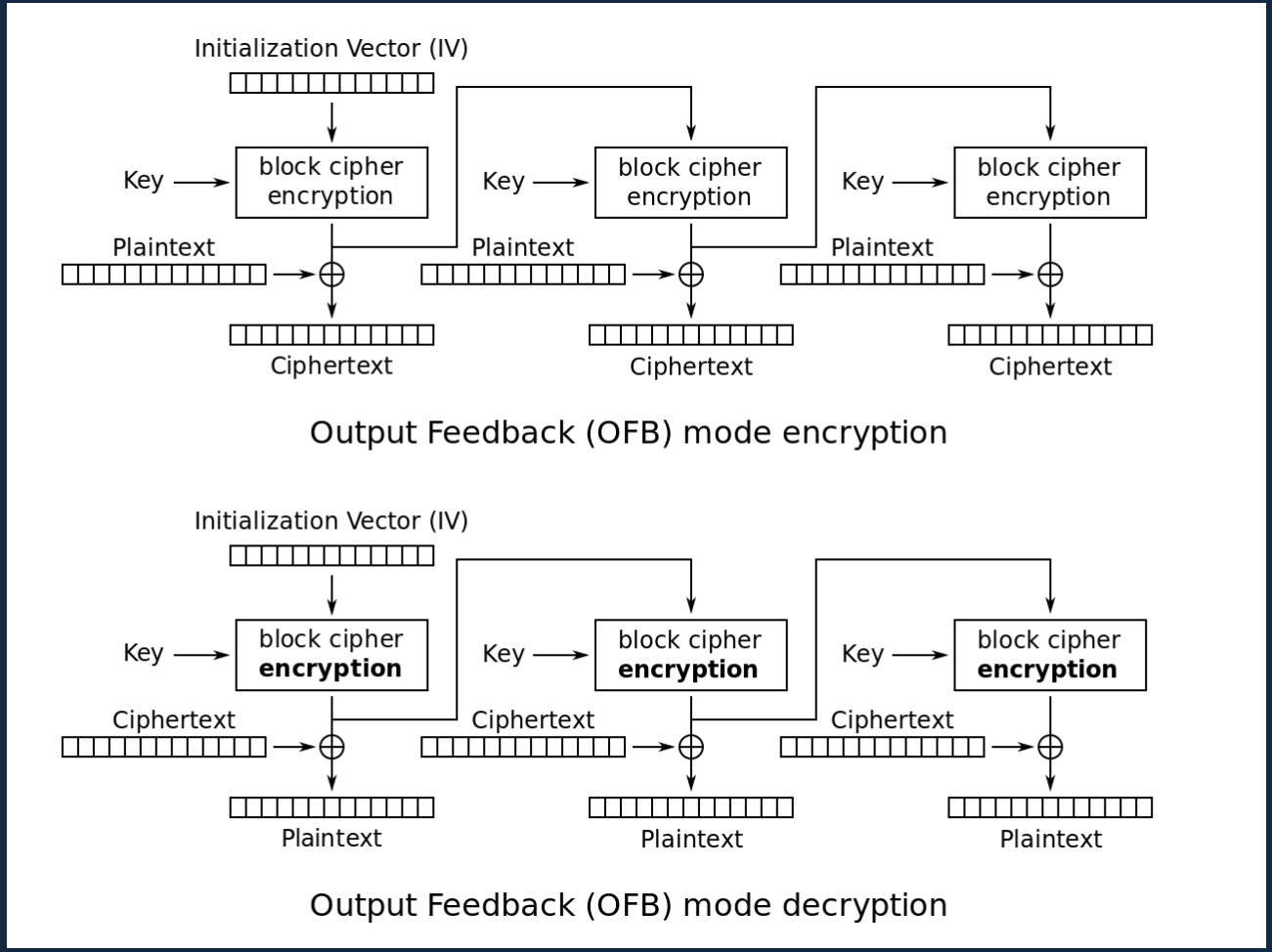
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Cipher Feedback



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

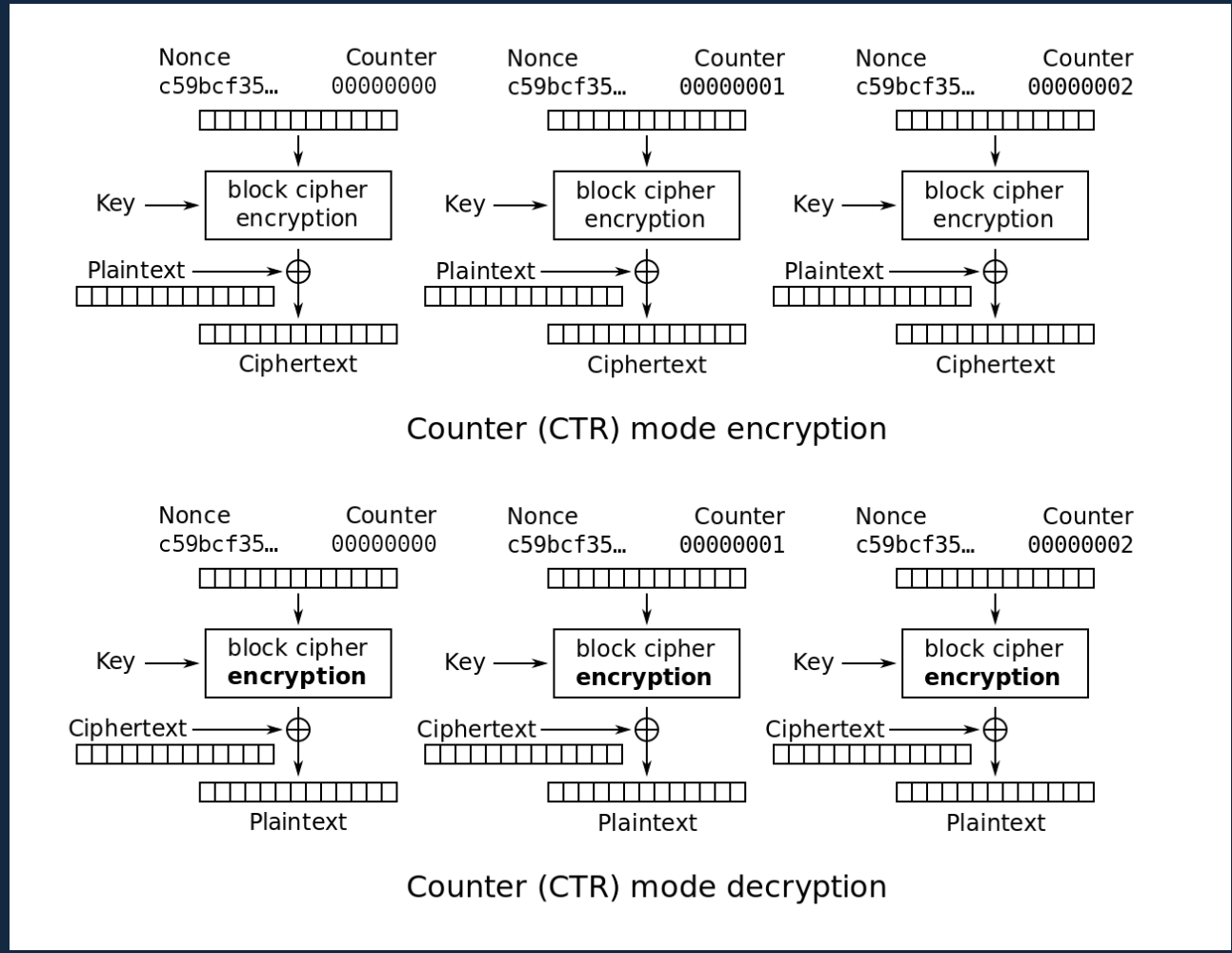
Output Feedback



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

AES

Counter



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Block Chiffren

- Block-Chiffren operieren fast immer mit einfachen einzelnen Operationen
- Diese werden hintereinander geschaltet
- Die meisten modernen Chiffren nutzen ARX (Addition-Rotation-XOR)
- Dies führt zu schnellen einfach implementierbaren Chiffren
- Zusätzlich ist jede Operation automatisch «constant-time» und verhindert so eine der Side-Channel-Attacks
- AES sowie auch DES verwenden zusätzlich noch S-Boxen (Substitutionsboxen) und P-Boxen (Permutierungsboxen)

AES

Initialisierungsvektor

ubique

Initialisierungsvektor (IV)

- Block-Ciphers können nur einzelne Blöcke verschlüsseln
- Wie bereits gesehen gibt es verschiedene Methoden diese Blöcke aneinander zu reihen
- Oft wird jedoch ein Initialzustand erwartet
- Dieser Zustand nennt sich Initialisierungsvektor
- Er sollte in den meisten Fällen eine Kryptografische-Nonce sein (also nur einmal verwendet werden)
- Für die Output-Feedback und Counter Modi muss er für jede Nachricht anderst sein, da sonst die «Zufallsbytes» immer die gleichen bleiben

AES

AES

ubique

AES

Advanced Encryption Standard

- AES ist eine Block-Chiffre
- Sie wurde ende 2000 als neuer Standard von NIST publiziert
- Die Chiffre gilt heute immer noch als Sicher
- Sie ist vom Amerikanischen Militär für Dokumente höchster Geheimhaltungsstufe zugelassen (AES-192 und AES-256)
- Ebenfalls wird sie vom Schweizer Militär zur Verschlüsselung von Dokumenten mit Klassifizierung (z.B. Geheim) benutzt (AES-128)

AES

Advanced Encryption Standard

- Die einzig bekannten Angriffe, reduzieren die Komplexität wenig unter die Komplexität von Ausprobieren

- Da mit 128 resp. 256 Bits gearbeitet wird, sind die Komplexitäten immer noch im Bereich der Lebzeit des Universums

- Da AES mittlerweile sehr Häufig benutzt wird, gibt es bestimmte Hardwareunterstützung

AES

Advanced Encryption Standard

- AES verwendet S-Boxen und Permutationen
- Die S-Box ersetzt jedes Element mit seinem «Multiplikativen Inversen» (ausser 0 was mit 0 ersetzt wird)
- Danach werden die Zeilen verschoben
- Anschliessend die Daten in einer Spalte mit einander vertauscht
- Am Ende jeder Runde wird ein Rundenschlüssel XOR’d

AES

AES-ECB

ubique

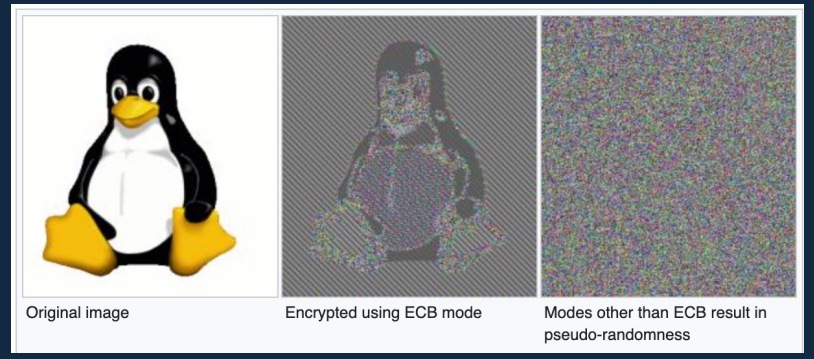
AES Electronic Code book

- Electronic Code Book Mode ist der einfachste Betriebsmodus

- Eine Nachricht wird in Blöcke aufgeteilt und jeder Block wird einzeln mit der Block-Chiffre verschlüsselt

- Dies führt allerdings dazu, dass gleiche Stellen gleich Verschlüsselt wird

- Entsprechend sollte diese Methode nicht verwendet werden



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_codebook_(ECB)

AES

AES-CBC

ubique

AES Cipher Block Chaining

- Im CBC Modus wird der Ciphertext der vorigen Verschlüsselung mit dem Klartext der nächsten XOR’d
- Dadurch wird mehr Zufall in den Ciphertext eingebaut
- Im ersten Schritt wird ein Initialisierungsvektor benötigt
- Falls IV nicht immer generiert wird, können Informationen über den ersten Block geleaked werden
- Ebenfalls können nur Nachrichten verschlüsselt werden, die eine Länge haben, die ein Vielfaches des Blocks ist
- Entsprechend muss der letzte Block «gepadded» werden
- Dadurch wird die Chiffre anfällig auf Padding-Oracle Attacken

AES

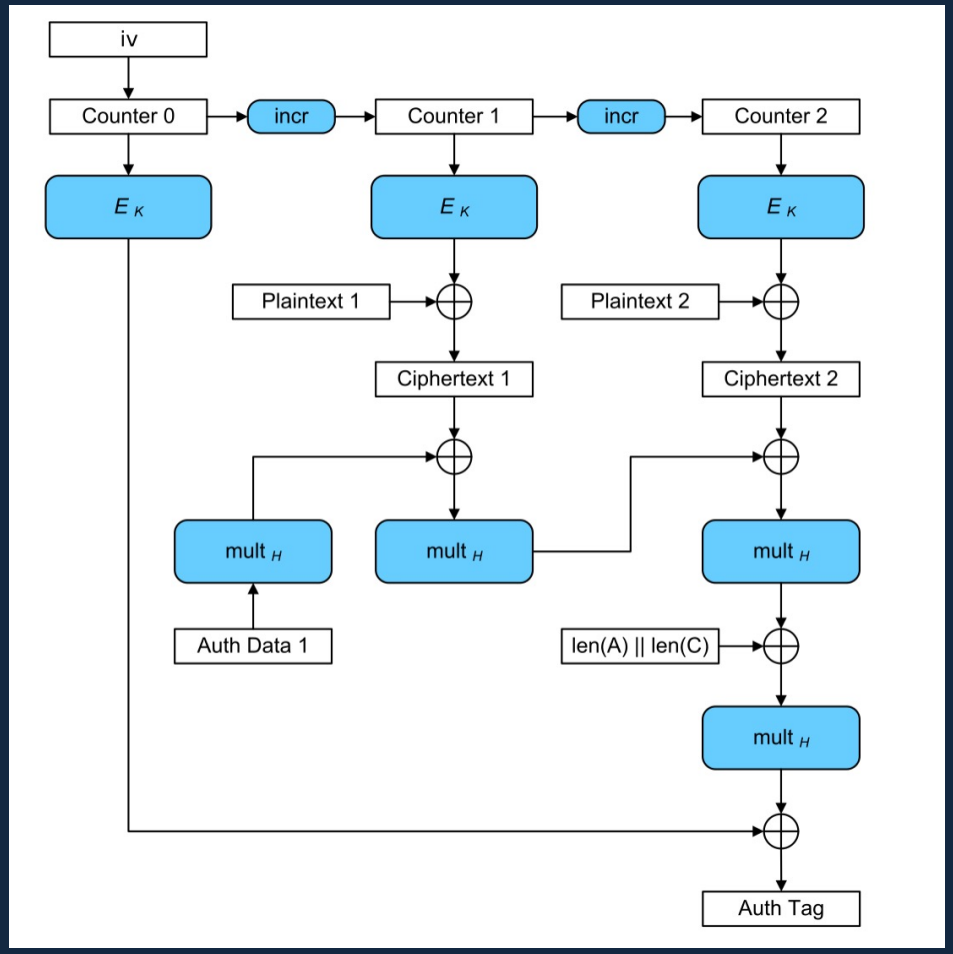
AES-GCM

ubique

AES Galois Counter Mode

- AES-GCM ist eine Cipher die ähnlich zum CTR Mode funktioniert
- Sie benutzt einen blockabhängigen Counter und einen IV um den Klartext zu verschlüsseln
- AES-GCM ist zusätzlich eine «authenticated-encryption» Methode
- Es wird in jeder Runde ein Message-Authentication-Code (MAC) berechnet
- Dieser erlaubt es einem Besitzer des Geheimschlüssels, sicherzustellen, dass keine Änderungen am Klartext bei der Übertragung durchgeführt wurde

AES Galois Counter Mode



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

AES

ChaCha20

ubique

AES

ChaCha20

- ChaCha20 wurde 2008 veröffentlicht

- Es ist eine Block-Chiffre die nur ARX-Operationen benutzt

- Sie hat insbesondere Vorteile auf Systemen, die keine Hardwareunterstützung für AES besitzen

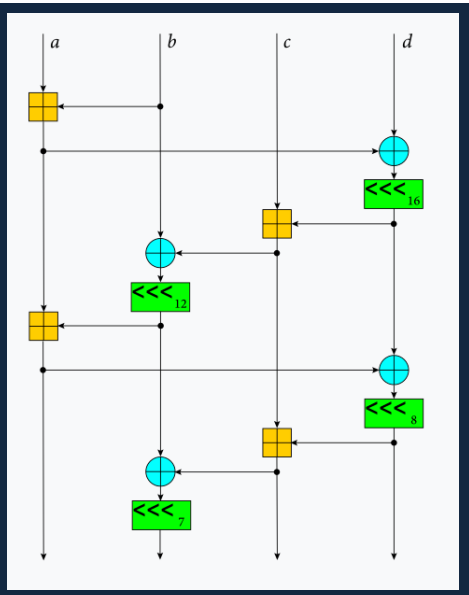
- Die Chiffre ist ebenfalls extrem einfach und resistent gegen Timing-Attacks (ARX Operationen sind alle Constant-Time)

AES

ChaCha20

- ChaCha20 besitzt einen internen Block bestehend aus 16 32bit Integers
- Eine Runde besteht aus vier Viertel-Runden
- Gerade Runden arbeiten auf Zeilen des internen Blocks
- Ungerade Runden arbeiten auf den Diagonalen des internen Blocks

"expa"	"nd 3"	"2-by"	"te k"
Key	Key	Key	Key
Key	Key	Key	Key
Counter	Nonce	Nonce	Nonce



AES

ChaCha20

```c
#include <stdint.h>
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
#define QR(a, b, c, d)(     \
    b ^= ROTL(a + d, 7),    \
    c ^= ROTL(b + a, 9),    \
    d ^= ROTL(c + b,13),    \
    a ^= ROTL(d + c,18))
#define ROUNDS 20

void salsa20_block(uint32_t out[16], uint32_t const in[16])
{
    int i;
    uint32_t x[16];

    for (i = 0; i < 16; ++i)
        x[i] = in[i];
    // 10 loops × 2 rounds/loop = 20 rounds
    for (i = 0; i < ROUNDS; i += 2) {
        // Odd round
        QR(x[ 0], x[ 4], x[ 8], x[12]); // column 1
        QR(x[ 5], x[ 9], x[13], x[ 1]); // column 2
        QR(x[10], x[14], x[ 2], x[ 6]); // column 3
        QR(x[15], x[ 3], x[ 7], x[11]); // column 4
        // Even round
        QR(x[ 0], x[ 1], x[ 2], x[ 3]); // row 1
        QR(x[ 5], x[ 6], x[ 7], x[ 4]); // row 2
        QR(x[10], x[11], x[ 8], x[ 9]); // row 3
        QR(x[15], x[12], x[13], x[14]); // row 4
    }
    for (i = 0; i < 16; ++i)
        out[i] = x[i] + in[i];
}
```

https://en.wikipedia.org/wiki/Salsa20#ChaCha_variant