

# Betriebsmodi bei Blockchiffren

**Dozent:** Prof. Dr. Michael Eichberg

**Version:** 1.0

**Basierend auf:** *Cryptography and Network Security - Principles and Practice, 8th Edition, William Stallings*



1

---

**Folien:**    **HTML:**    <https://delors.github.io/sec-blockchiffre-operationsmodi/folien.de.rst.html>  
              **PDF:**        <https://delors.github.io/sec-blockchiffre-operationsmodi/folien.de.rst.html.pdf>  
**Fehler melden:**  
              <https://github.com/Delors/delors.github.io/issues>

# Betriebsmodi

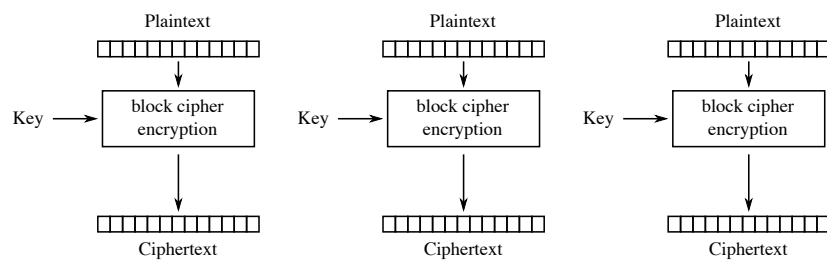
- Eine Technik zur Verbesserung der Wirkung eines kryptografischen Algorithmus oder zur Anpassung des Algorithmus an ein Anwendungsszenario. Insbesondere in Abhängigkeit von der Länge des Klartexts.
- Um eine Blockchiffre in einer Vielzahl von Anwendungen einsetzen zu können, hat das NIST fünf Betriebsmodi definiert.
  - Die fünf Modi decken eine breite Palette von Verschlüsselungsanwendungen ab, für die eine Blockchiffre verwendet werden kann.
  - Diese Modi sind für die Verwendung mit jeder symmetrischen Blockchiffre vorgesehen, einschließlich 3DES und AES.

# Betriebsmodi - Übersicht

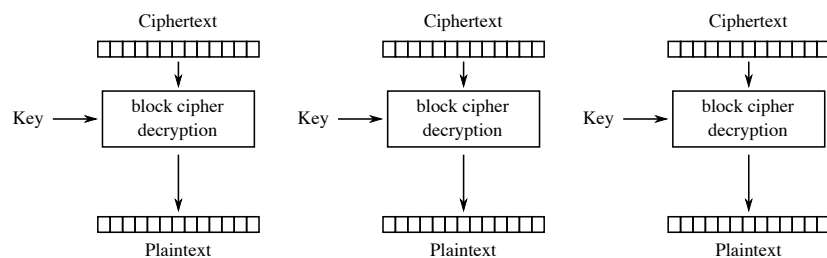
Modus	Beschreibung	Typische Anwendung
<b>Electronic Codebook (ECB)</b>	Jeder Block von Klartextbits wird unabhängig voneinander mit demselben Schlüssel verschlüsselt.	<ul style="list-style-type: none"> <li>■ Sichere Übertragung einzelner Werte (z. B. eines Verschlüsselungsschlüssels)</li> </ul>
<b>Cipher Block Chaining (CBC)</b>	Die Eingabe für den Verschlüsselungsalgorithmus ist die XOR-Verknüpfung des nächsten Klartextblocks mit dem vorangegangenen Chiffretextblock.	<ul style="list-style-type: none"> <li>■ Universelle blockorientierte Übertragung</li> <li>■ Authentifizierung</li> </ul>
<b>Cipher Feedback (CFB)</b>	Die Eingabe wird Bit für Bit verarbeitet. Der vorhergehende Chiffretext wird als Eingabe für den Verschlüsselungsalgorithmus verwendet, um eine pseudozufällige Ausgabe zu erzeugen, die mit dem Klartext XOR-verknüpft wird, um die nächste Einheit des Chiffretextes zu erzeugen.	<ul style="list-style-type: none"> <li>■ Allgemeine stromorientierte Übertragung</li> <li>■ Authentifizierung</li> </ul>
<b>Output Feedback (OFB)</b>	Ähnlich wie CFB, mit dem Unterschied, dass die Eingabe für den Verschlüsselungsalgorithmus die vorangegangene Verschlüsselungsausgabe ist, und volle Blöcke verwendet werden.	<ul style="list-style-type: none"> <li>■ Stromorientierte Übertragung über verrauschte Kanäle (z. B. Satellitenkommunikation)</li> </ul>
<b>Counter (CTR)</b>	Jeder Klartextblock wird mit einem verschlüsselten Zähler XOR-verknüpft. Der Zähler wird für jeden nachfolgenden Block erhöht.	<ul style="list-style-type: none"> <li>■ Blockorientierte Übertragung für allgemeine Zwecke</li> <li>■ Nützlich für Hochgeschwindigkeitsanforderungen</li> </ul>

# Grundlegende Blockchiffren

# Electronic Codebook



Electronic Codebook (ECB) mode encryption



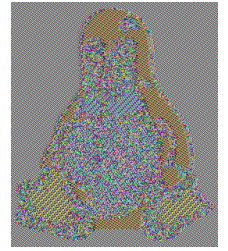
Electronic Codebook (ECB) mode decryption

Autor: <https://commons.wikimedia.org/wiki/User:WhiteTimberwolf>

# Probleme bei der Verwendung der Verschlüsselung im ECB-Modus

*ECB-Tux* - der Linux-Pinguin verschlüsselt im ECB-Modus:

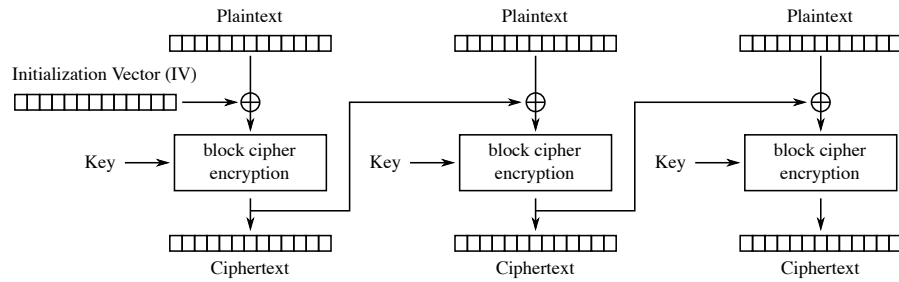
Quelle: <https://github.com/robertdavidgraham/ecb-penguin>



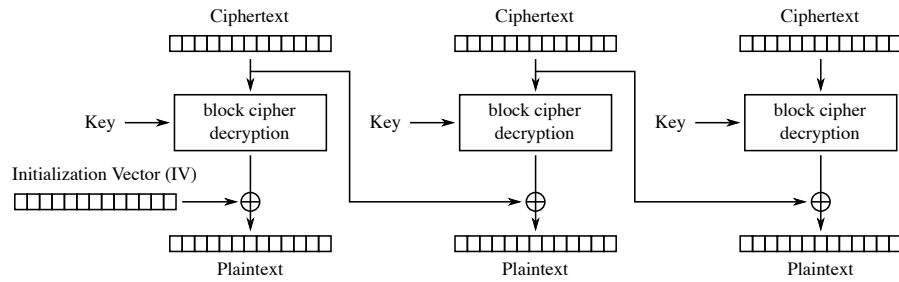
Kriterien und Eigenschaften für die Bewertung und Konstruktion von Blockchiffre-Betriebsarten, die ECB überlegen sind.

- Overhead
- Fehlerbehebung
- Fehlerfortpflanzung
- Streuung
- Sicherheit

# Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Autor: <https://commons.wikimedia.org/wiki/User:WhiteTimberwolf>

**Blockchiffren, die als Stromchiffren  
verwendet werden können.**



# Konvertierung von Blockchiffren in Stromchiffre


Bei AES, DES oder jeder anderen Blockchiffre erfolgt die Verschlüsselung immer Block-für-Block mit Blockgrößen von  $b$  Bits:

- Im Fall von (3)DES:  $b = 64$
- Im Fall von AES:  $b = 128$

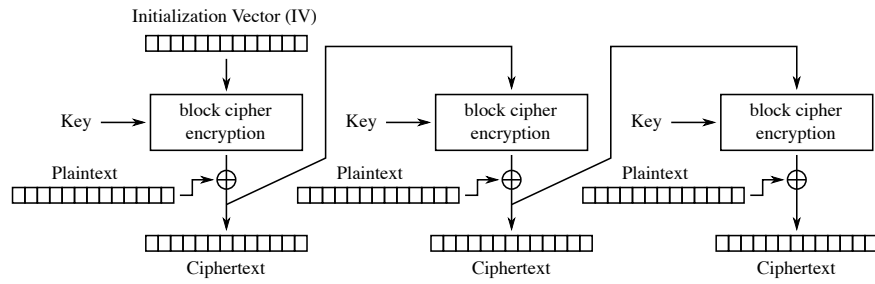
## Hinweis

Es gibt drei Modi, die es ermöglichen, eine Blockchiffre in eine zeichenorientierte Stromchiffre umzuwandeln:

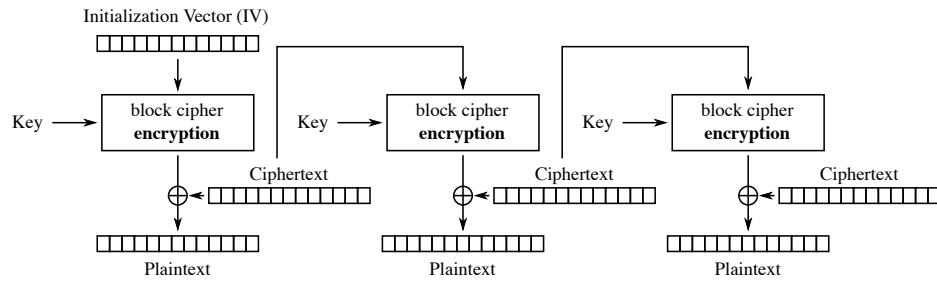
- Cipher Feedback Mode (CFB)
- Output Feedback Mode (OFB)
- Counter Mode (CTR)

D. h., es ist kein Auffüllen (  *Padding*) erforderlich, wenn die Nachricht nicht ein Vielfaches der Blockgröße ist.

# Cipher Feedback Mode



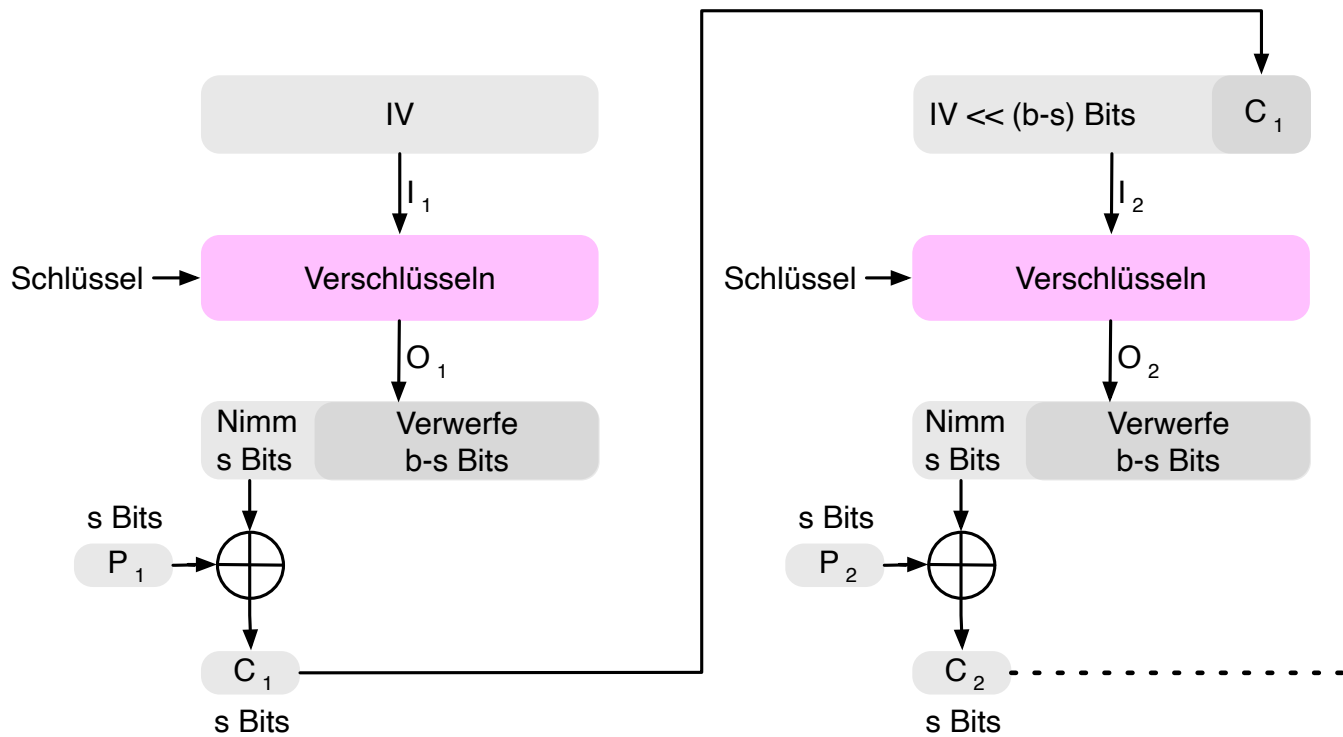
Cipher Feedback (CFB) mode encryption



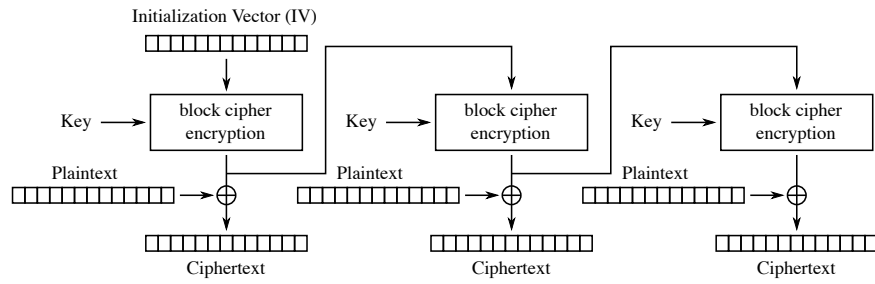
Cipher Feedback (CFB) mode decryption

Autor: <https://commons.wikimedia.org/wiki/User:WhiteTimberwolf>

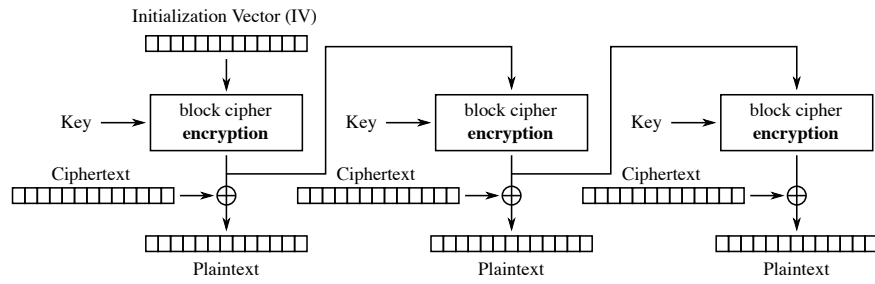
# Cipher Feedback Mode als Stromchiffre



# Output Feedback Mode



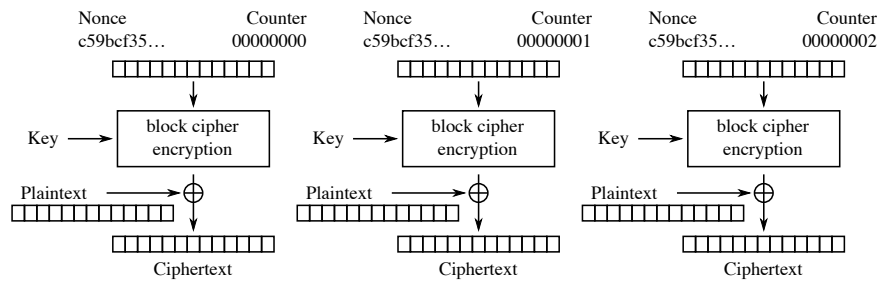
Output Feedback (OFB) mode encryption



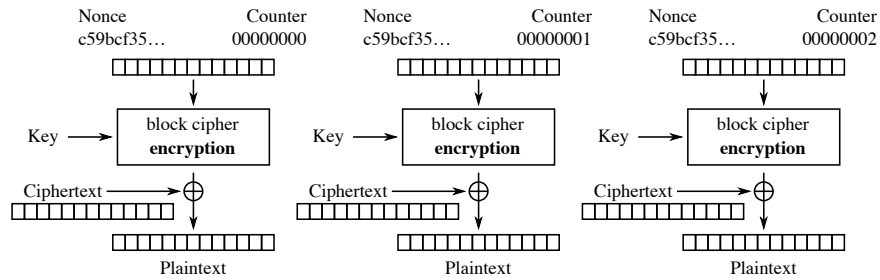
Output Feedback (OFB) mode decryption

Autor: <https://commons.wikimedia.org/wiki/User:WhiteTimberwolf>

# Counter Mode



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Autor: <https://commons.wikimedia.org/wiki/User:WhiteTimberwolf>

# **Counter Mode - Vorteile**

## **Hardware-Effizienz:**

kann von der Parallelisierung der Hardware profitieren

## **Software-Effizienz:**

leicht parallelisierbar in Software

## **Vorverarbeitung:**

die Verschlüsselung der Zähler

## **Zufälliger Zugriff:**

der i-te Block des Klartextes/des Chiffretextes kann im Zufallszugriff verarbeitet werden

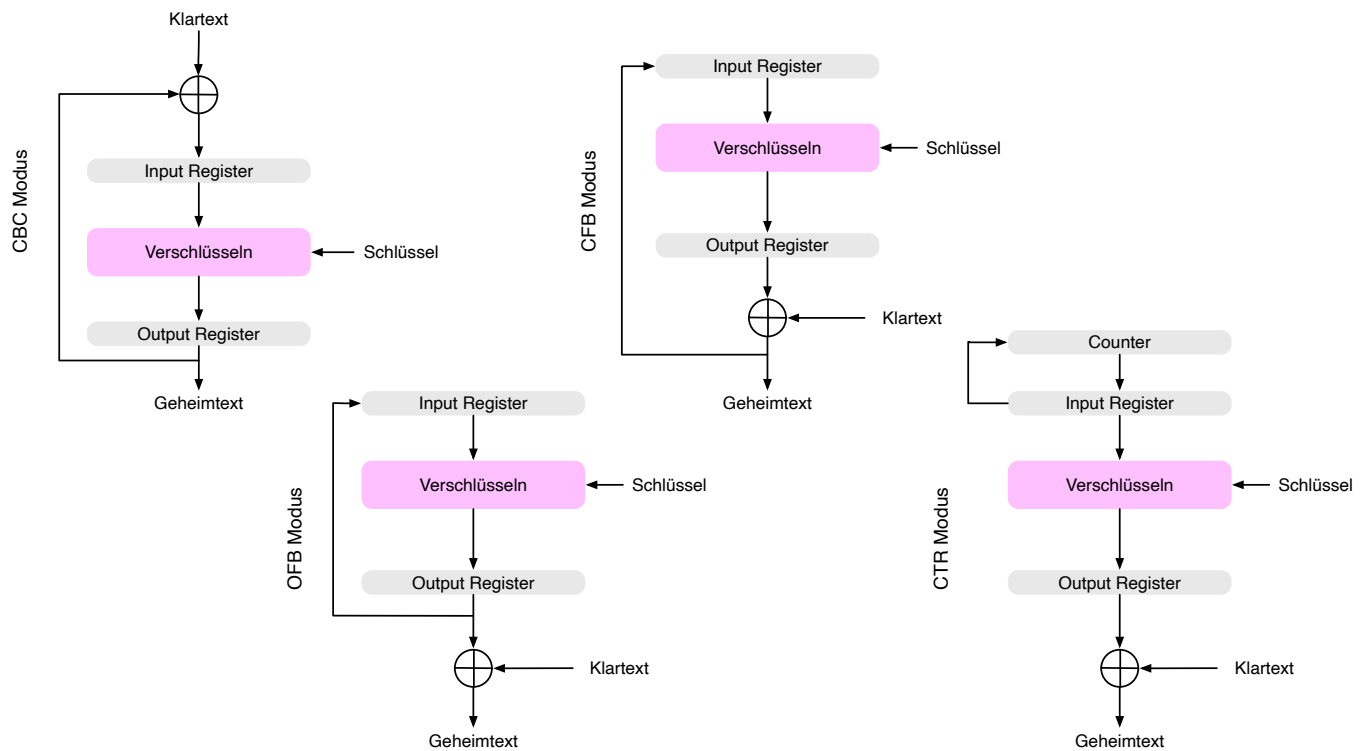
## **Nachweisbare Sicherheit:**

genauso sicher wie die anderen Verfahren

## **Einfachheit:**

es wird nur der Verschlüsselungsalgorithmus benötigt

# Rückkopplungseigenschaften[1] der Betriebsmodi



[1] (Feedback Characteristics)

## **Spezielle Betriebsmodi**



# XTS-AES Modus für blockorientierte Speichergeräte

2010 vom NIST als zusätzlicher Blockchiffre-Betriebsmodus genehmigt.

Modus ist auch ein IEEE-Standard, IEEE Std 1619-2007

- Die Norm beschreibt eine Verschlüsselungsmethode für Daten, die in sektorbasierten Geräten gespeichert sind, wobei das Bedrohungsmodell einen möglichen Zugriff des Gegners auf die gespeicherten Daten beinhaltet.
- Hat breite Unterstützung der Industrie erhalten.

## Frage

Welche potenziellen Bedrohungen sind relevant?

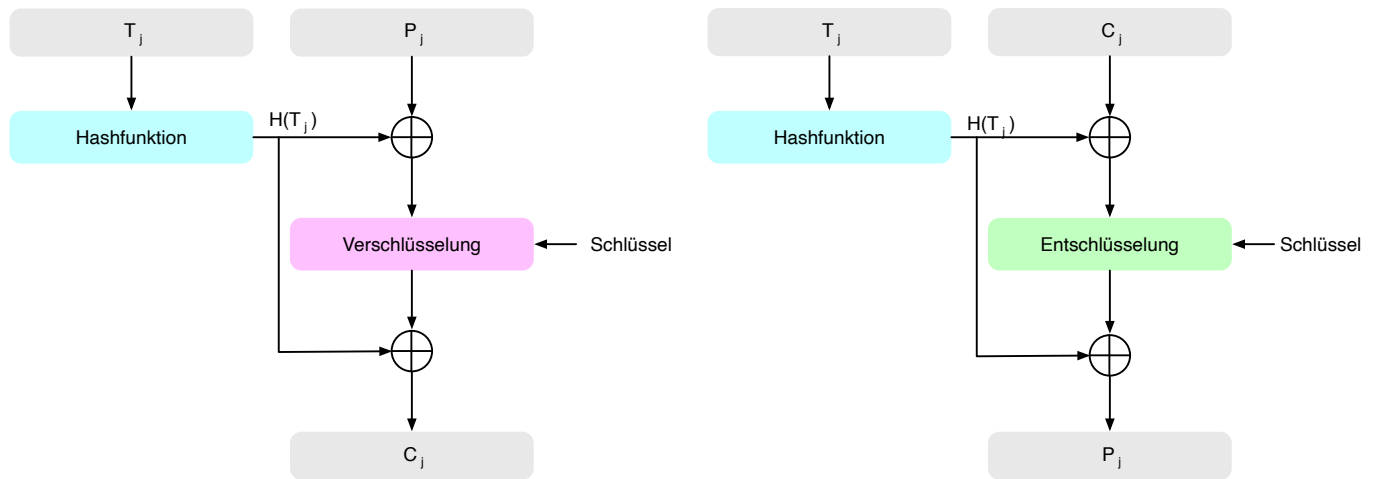
# ***Tweakable* Blockchiffren - Bestandteile**

- Der XTS-AES-Modus basiert auf dem Konzept einer veränderbaren (🚩 *tweakable*) Blockchiffre.
- Allgemeine Struktur:

Um den Chiffriertext zu berechnen, wird benötigt:

  - **Klartext**
  - **Symmetrischer Schlüssel**
  - **Tweak**
- Der *Tweak* muss nicht geheim gehalten werden; der Zweck ist, Variabilität zu bieten.

# ***Tweakable* Blockchiffren - grundlegende Struktur**



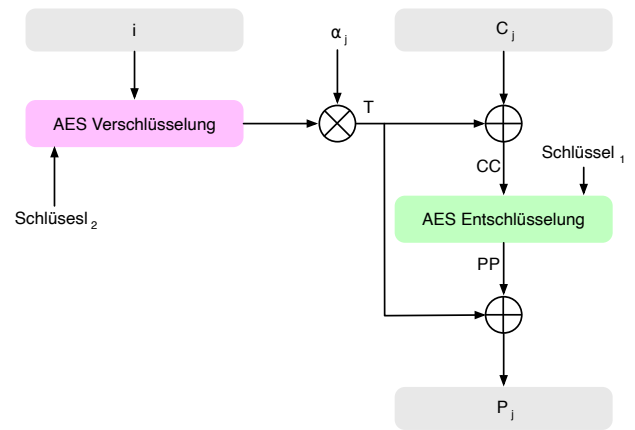
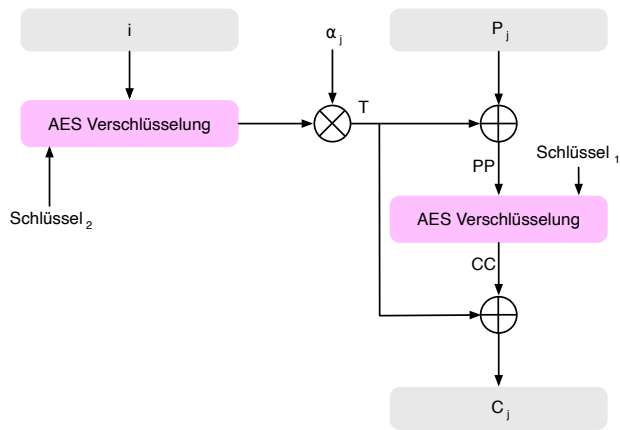
# Anforderungen an die Speicherverschlüsselung

Die Anforderungen an die Verschlüsselung gespeicherter Daten, die auch als *data at rest* bezeichnet werden, unterscheiden sich von denen für übertragene Daten.

Die Norm P1619 wurde in Hinblick auf folgende Eigenschaften entwickelt:

- Der Chiffretext ist für einen Angreifer frei verfügbar.
- Das Datenlayout wird auf dem Speichermedium und beim Transport nicht verändert.
- Der Zugriff auf die Daten erfolgt in Blöcken fester Größe und unabhängig voneinander.
- Die Verschlüsselung erfolgt in 16-Byte-Blöcken, die unabhängig voneinander sind.
- Es werden keine weiteren Metadaten verwendet, außer der Position der Datenblöcke innerhalb des gesamten Datensatzes.
- Derselbe Klartext wird an verschiedenen Stellen in verschiedene Chiffretexte verschlüsselt, aber immer in denselben Chiffretext, wenn er wieder an dieselbe Stelle geschrieben wird.
- Ein standardkonformes Gerät kann für die Entschlüsselung von Daten konstruiert werden, die von einem anderen standardkonformen Gerät verschlüsselt wurden.

# XTS-AES Operation auf einem Block



- Schlüssel: es gilt:  $Schlüssel = Schlüssel_1 || Schlüssel_2$
- $P_j$ : Der  $j$ -te Block des Klartexts. Alle Blöcke haben eine Länge von 128 bits. Eine Klartextdateneinheit – in der Regel ein Festplattensektor – besteht aus einer Folge von Klartextblöcken.
- $C_j$ : Der  $j$ -te Block des Chiffretextes.
- $j$ : Die fortlaufende Nummer des 128-Bit-Blocks innerhalb der Dateneinheit.
- $i$ : Der Wert des 128-Bit-Tweaks.
- $\alpha$ : Ein primitives Element des  $GF(2^{128})$  welches dem Polynom  $x$  (d. h. 0000...0010) entspricht.
- $\alpha^j$ :  $\alpha$   $j$  mal mit sich selbst multipliziert im Körper  $GF(2^{128})$
- $\oplus$  Bitwise XOR
- $\otimes$  Modulare Multiplikation mit Binärkoeffizienten modulo  $x^{128} + x^7 + x^2 + x + 1$ .

- Warum ist es bei CBC wichtig, den Initialisierungsvektor (IV) zu schützen?
- In welchen Betriebsarten ist eine Auffüllung (🇺🇸 *Padding*) notwendig?
- Was geschieht im Falle eines Übertragungsfehlers (einzelner Bitflip im Chiffretext) bei ECB, CBC, CFB, OFB, CTR?
- Warum muss der IV im Falle von OFB eine Nonce (🇺🇸 *Number used ONCE*) sein (d. h. eine Zahl, die nur einmal für die Ausführung des Verschlüsselungsalgorithmus verwendet wird)?
- Sie möchten feststellen, ob ein Programm zur Verschlüsselung von Dateien den ECB-Modus verwendet. Was müssen Sie tun?

Warum ist es bei CBC wichtig, den Initialisierungsvektor (IV) zu schützen?

In welchen Betriebsarten ist eine Auffüllung (🚧 *Padding*) notwendig?



Was geschieht im Falle eines Übertragungsfehlers (einzelner Bitflip im Chiffretext) bei ECB, CBC, CFB, OFB, CTR?

Warum muss der IV im Falle von OFB eine Nonce (🚩 *Number used ONCE*) sein (d. h. eine Zahl, die nur einmal für die Ausführung des Verschlüsselungsalgorithmus verwendet wird)?

Sie möchten feststellen, ob ein Programm zur Verschlüsselung von Dateien den ECB-Modus verwendet. Was müssen Sie tun?

Verwenden Sie den OFB-Modus in Kombination mit einer Caesar-Chiffre. Die Blockgröße ist ein einzelnes Zeichen. Der Schlüssel ist die Anzahl der Zeichen, um die Sie ein Zeichen verschieben wollen - wie zuvor. Die IV ist ein Zeichen. Damit sie ein XOR durchführen können, ordnen wir jedem Zeichen einen Wert zu und erweitern das Alphabet um die Ziffern 1 bis 3, "!", "?" und das "\_". Auf diese Weise ist es immer möglich, ein sinnvolles Zeichen auszugeben.

Daraus ergibt sich die folgende Kodierung:

Index	Zeichen	Binärdarstellung
0	A	00000
1	B	00001
2	C	00010
3	D	00011
4	E	00100
5	F	00101
6	G	00110
7	H	00111
8	I	01000
9	J	01001
10	K	01010

Index	Zeichen	Binärdarstellung
11	L	01011
12	M	01100
13	N	01101
14	O	01110
15	P	01111
16	Q	10000
17	R	10001
18	S	10010
19	T	10011
20	U	10100
21	V	10101

Index	Zeichen	Binärdarstellung
22	W	10110
23	X	10111
24	Y	11000
25	Z	11001
26	1	11010
27	2	11011
28	3	11100
29	!	11101
30	?	11110
31	_	11111

Verschlüsseln Sie nun einige Nachrichten mit dieser Chiffre. Welchen Effekt hat die Anwendung des OFB-Modus auf die Nachrichten?

Verwenden Sie den OFB-Modus in Kombination mit einer Caesar-Chiffre. Die Blockgröße ist ein einzelnes Zeichen. Der Schlüssel ist die Anzahl der Zeichen, um die Sie ein Zeichen verschieben wollen - wie zuvor. Die IV ist ein Zeichen. Damit sie ein XOR durchführen können, ordnen wir jedem Zeichen einen Wert zu und erweitern das Alphabet um die Ziffern 1 bis 3, "!", "?" und das "\_". Auf diese Weise ist es immer möglich, ein sinnvolles Zeichen auszugeben.

Daraus ergibt sich die folgende Kodierung:

Index	Zeichen	Binärdarstellung
0	A	00000
1	B	00001
2	C	00010
3	D	00011
4	E	00100
5	F	00101
6	G	00110
7	H	00111
8	I	01000
9	J	01001
10	K	01010

Index	Zeichen	Binärdarstellung
11	L	01011
12	M	01100
13	N	01101
14	O	01110
15	P	01111
16	Q	10000
17	R	10001
18	S	10010
19	T	10011
20	U	10100
21	V	10101

Index	Zeichen	Binärdarstellung
22	W	10110
23	X	10111
24	Y	11000
25	Z	11001
26	1	11010
27	2	11011
28	3	11100
29	!	11101
30	?	11110
31	_	11111

Verschlüsseln Sie nun einige Nachrichten mit dieser Chiffre. Welchen Effekt hat die Anwendung des OFB-Modus auf die Nachrichten?