Betriebsmodi bei Blockchiffren

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de
Version: 1.3.3

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

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

Kontrollfragen: https://delors.github.io/sec-blockchiffre-operationsmodi/kontrollfragen.-de.rst.html

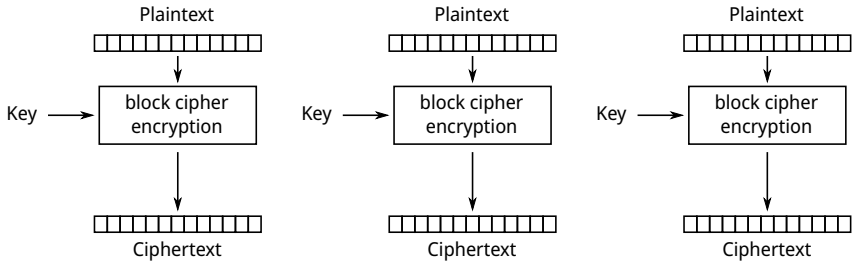
Fehler melden: https://github.com/Delors/delors.github.io/issues

Betriebsmodi

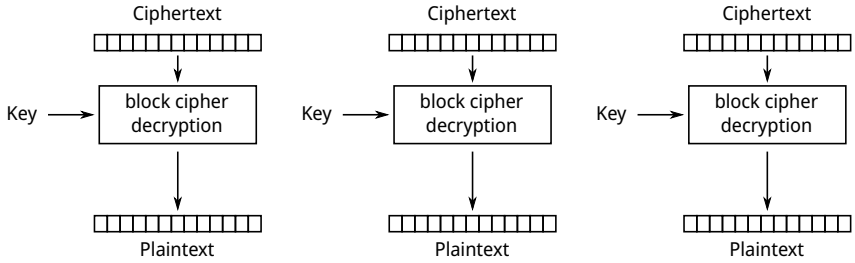
- Eine Technik zur Verbesserung der Wirkung eines kryptografischen Algorithmus oder zur Anpassung des Algorithmus an ein Anwendungsszenario. Insbesondere in Abhängigkeit von der Länge des Klartexts.
- Um eine Blockchiffre in einer Vielzahl von Anwendungen einsetzen zu können, hat das NIST fünf Betriebsmodi 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.

1. Grundlegende Blockchiffren

Electronic Codebook[1]



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

---

[1] Bilder von: White Timberwolf

Padding-Modi in Blockchiffren

Bemerkung

Bei Blockchiffren (z. B. AES mit 128 Bit = 16 Byte Blockgröße) muss die zu verschlüsselnde Nachricht eine exakte Vielfache der Blockgröße sein. Ist das nicht der Fall, wird Padding verwendet, um die Nachricht auf ein vielfaches der Blockgröße zu bringen.

PKCS#7

Füllt den restlichen Block mit Bytes, deren Wert gleich der Anzahl hinzugefügter Bytes ist.

Beispiel

- Nachricht mit 13 Byte → 3 Bytes Padding → `03 03 03`
- Nachricht mit Blocklänge (z. B. 16 Byte) → ein kompletter zusätzlicher Block mit 16 × `0x10`

ANSI X.923

Auffüllen mit `0x00`, außer dem letzten Byte, das die Anzahl Padding-Bytes angibt.

ISO/IEC 7816-4

Beginnt mit `0x80`, gefolgt von Nullen.

Zero Padding

Füllt mit Nullen (`0x00`) auf.

Achtung!

Funktioniert *nur*, wenn die Nachricht nie mit `0x00` endet, sonst ist Entschlüsselung mehrdeutig!

Verhalten bei voller Blockgröße - Zusammenfassung

Angenommen, die Nachricht ist exakt 16 Byte lang (z. B. "*1234567890ABCDEF*"), so ergibt sich:

Padding-Modus	Erweiterte Nachricht (hex)
PKCS#7	... 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
ANSI X.923	... 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10
ISO/IEC 7816-4	... 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Zero Padding	... 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

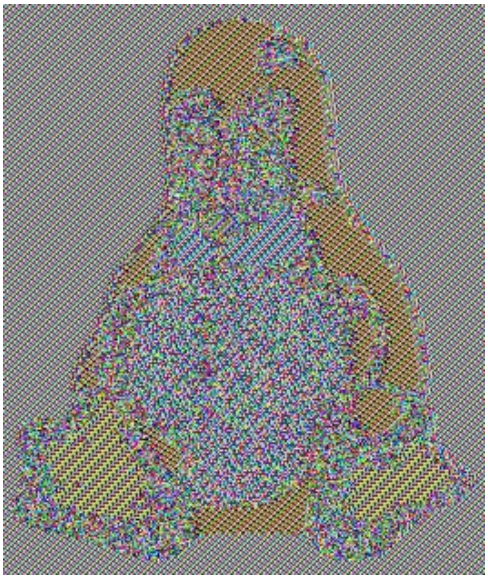
Hinweis

Bei allen Modi wird bei Nachrichten mit dem x-fachen der Blocklänge ein **zusätzlicher** Padding-Block hinzugefügt, um bei der Entschlüsselung korrekt erkennen zu können, ob Padding entfernt werden muss.

Probleme bei der Verwendung der Verschlüsselung im ECB-Modus

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

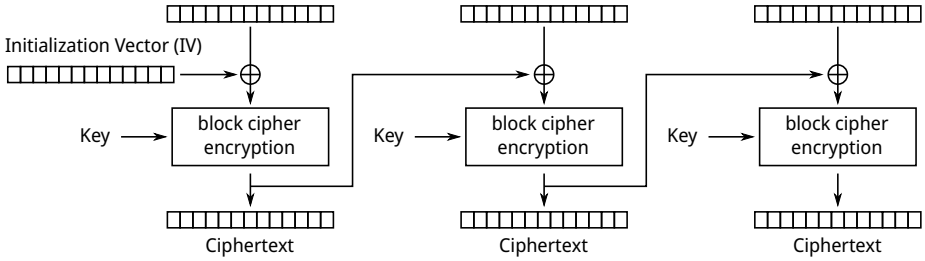
Quelle: https://github.-com/robertdavidgra-ham/ecb-penguin



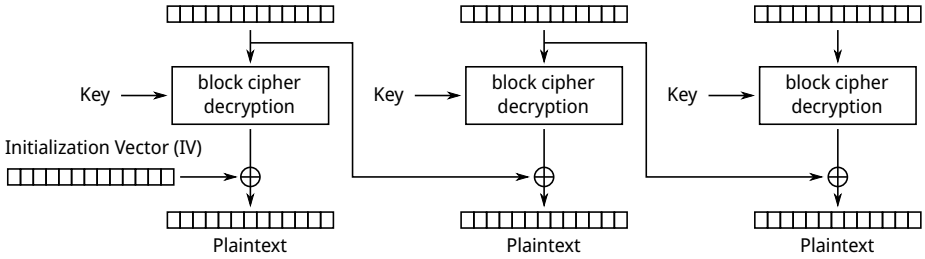
Kriterien und Eigenschaften für die Bewertung und Konstruktion von Blockchiffre-Betriebsarten, die ECB überlegen sind.

- Overhead
- Fehlerbehebung
- Fehlerfortpflanzung
- Streuung
- Sicherheit

Cipher Block Chaining[2]



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

---

[2] Bilder von: White Timberwolf

2. Blockchiffren, die als Stromchiffren verwendet werden können.

Konvertierung von Blockchiffren in Stromchiffre

Hinweis

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

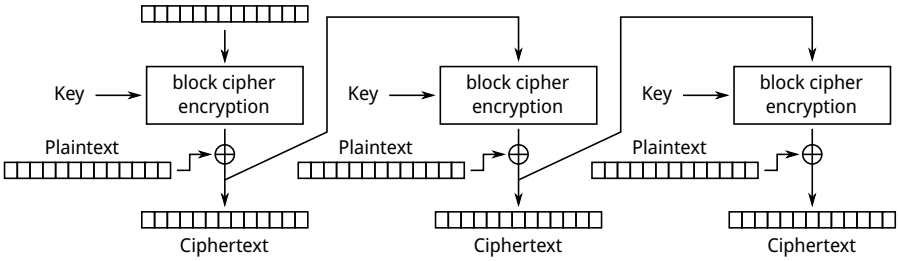
- Cipher Feedback Mode (CFB)
- Output Feedback Mode (OFB)
- Counter Mode (CTR)

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

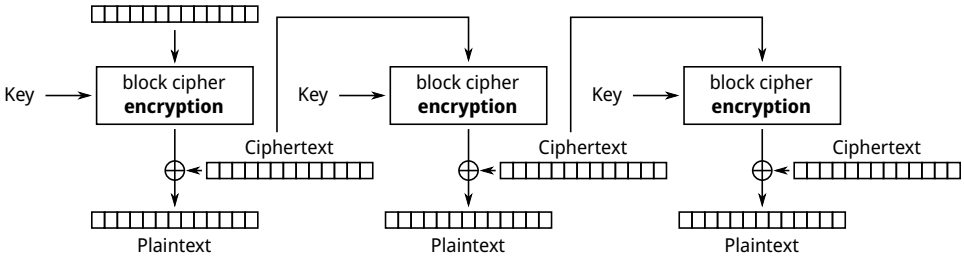
Bei AES, DES oder jeder anderen Blockchiffre erfolgt die Verschlüsselung immer Block-für-Block mit Blockgrößen von b Bits:

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

Cipher Feedback Mode[3]



Cipher Feedback (CFB) mode encryption

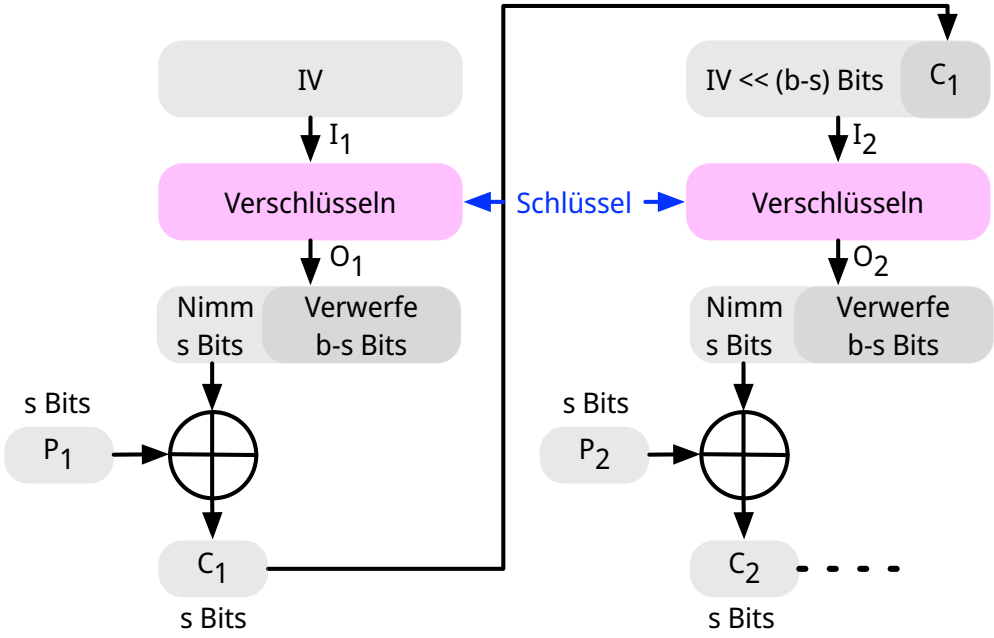


Cipher Feedback (CFB) mode decryption

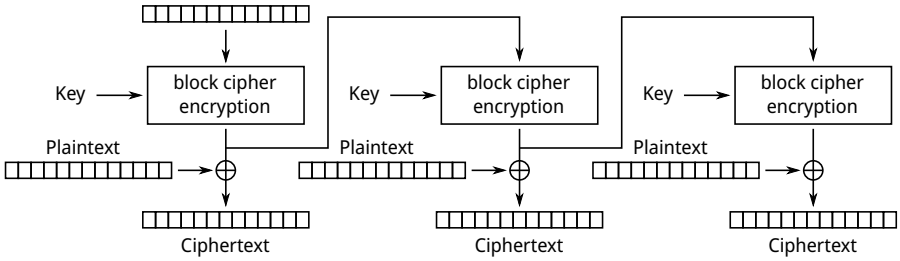
---

[3] Bilder von: White Timberwolf

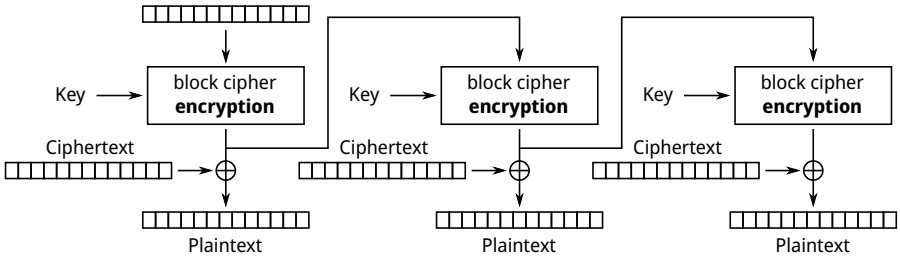
Cipher Feedback Mode als Stromchiffre



Output Feedback Mode



Output Feedback (OFB) mode encryption

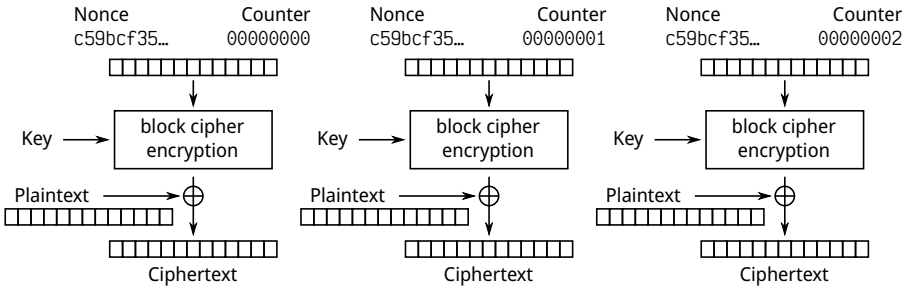


Output Feedback (OFB) mode decryption

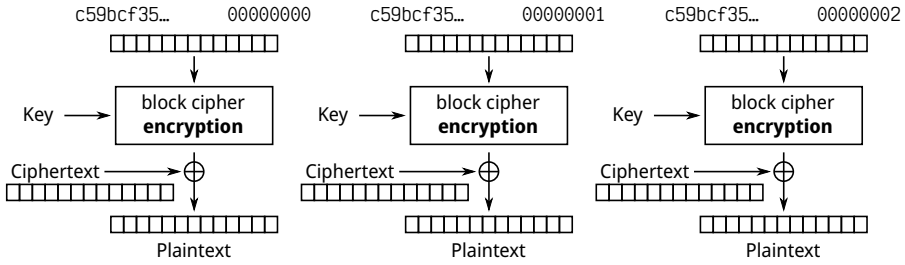
---

[4] Bilder von: White Timberwolf

Counter Mode



Counter (CTR) mode encryption



Counter (CTR) mode decryption

---

[5] Bilder von: White Timberwolf

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

Betriebsmodi - Übersicht

Modus	Beschreibung	Typische Anwendung
Electronic Codebook (ECB)	Jeder Block von Klartextbits wird unabhängig voneinander mit demselben Schlüssel verschlüsselt.	■ Sichere Übertragung einzelner Werte (z. B. eines Verschlüsselungsschlüssels)
Cipher Block Chaining (CBC)	Die Eingabe für den Verschlüsselungsalgorithmus ist die XOR-Verknüpfung des nächsten Klartextblocks mit dem vorangegangenen Chiffretextblock.	■ Universelle blockorientierte Übertragung ■ Authentifizierung
Cipher Feedback (CFB)	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.	■ Allgemeine stromorientierte Übertragung ■ Authentifizierung
Output Feedback (OFB)	Ähnlich wie CFB, mit dem Unterschied, dass die Eingabe für den Verschlüsselungsalgorithmus die vorangegangene Verschlüsselungsausgabe ist, und volle Blöcke verwendet werden.	■ Stromorientierte Übertragung über verrauschte Kanäle (z. B. Satellitenkommunikation)
Counter (CTR)	Jeder Klartextblock wird mit einem verschlüsselten Zähler XOR-verknüpft. Der Zähler wird für jeden nachfolgenden Block erhöht.	■ Blockorientierte Übertragung für allgemeine Zwecke ■ Nützlich für Hochgeschwindigkeitsanforderungen

3. Spezielle Betriebsmodi

XTS-AES Modus für block-orientierte Speichergeräte

2010 vom NIST als zusätzlicher Blockchiffre-Betriebsmodus genehmigt.

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

> **Frage**
>
> Welche potenziellen Bedrohungen sind relevant?

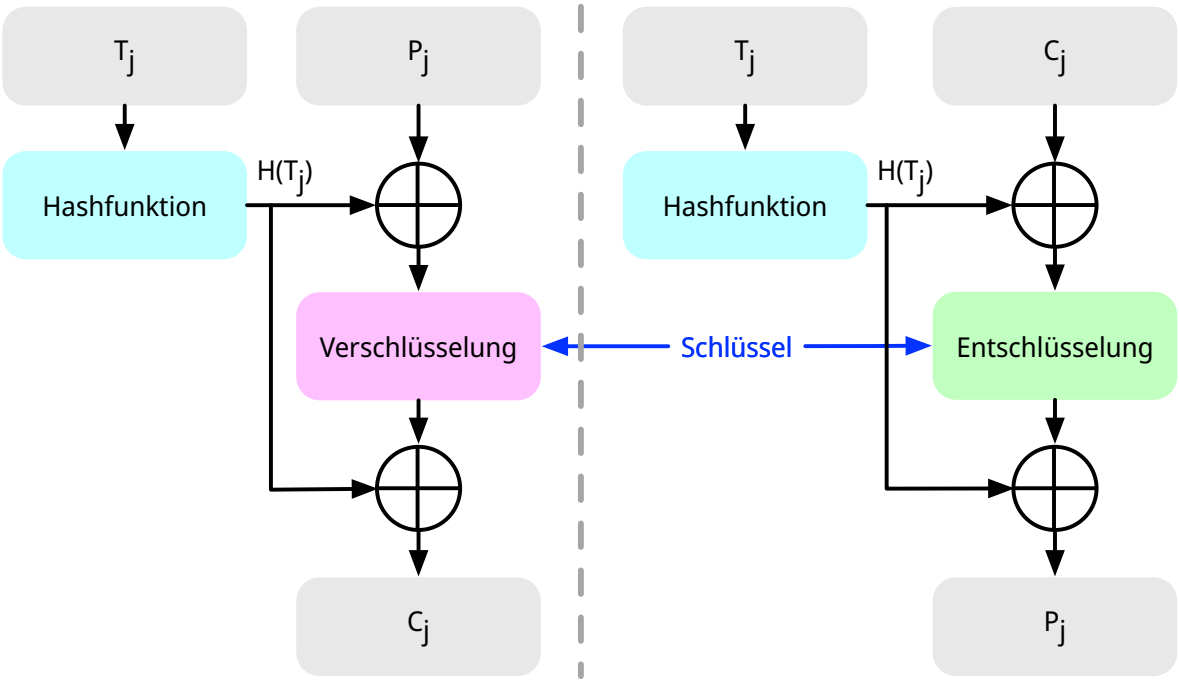
- Die Norm beschreibt eine Verschlüsselungsmethode für Daten, die in sektor-basierten Geräten gespeichert sind, wobei das Bedrohungsmodell einen möglichen Zugriff des Gegners auf die gespeicherten Daten beinhaltet.
- Hat breite Unterstützung der Industrie erhalten.

Tweakable Blockchiffren - Bestandteile

- Der XTS-AES-Modus basiert auf dem Konzept einer veränderbaren (🇺🇸 *tweakable*) Blockchiffre.
- Um den Chiffriertext zu berechnen, wird benötigt:
  - **Klartext**
  - **Symmetrischer Schlüssel**
  - **Tweak**
- Der *Tweak* muss nicht geheim gehalten werden; der Zweck ist, Variabilität zu bieten.

Ein Tweak ist insbesondere bei der Verschlüsselung von Daten auf Speichergeräten wichtig, da der gleiche Klartext an verschiedenen Stellen in verschiedene Chiffretexte verschlüsselt wird, aber immer in denselben Chiffretext, wenn er wieder an dieselbe Stelle geschrieben wird. Ein Tweak ist ein Modifikator, der für die unterschiedliche Verarbeitung gleicher Daten sorgt.

Tweakable Blockchiffren - grundlegende Struktur



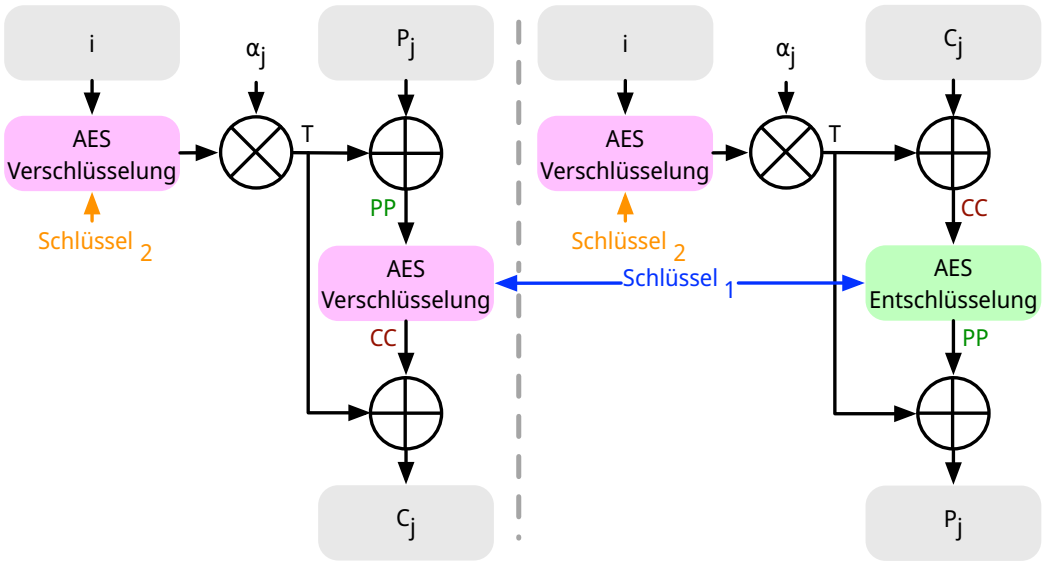
Anforderungen an die Speicherverschlüsselung

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

Die IEEE 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



※ Hinweis

Dargestellt ist der Fall, dass alle zu verschlüsselnden (Klartext-)Blöcke eine Größe von 128 Bit haben. Sollt der letzte Block nicht eine Größe von 128 Bit haben, dann kommt „Cipher Text Stealing“ zum Einsatz.

- 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 (Klartext)dateneinheit – 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.
- α: Ein primitives Element des GF(2^{128}) welches dem Polynom x (d. h. $0000...0010_b$) entspricht.
- α^j: α j mal mit sich selbst multipliziert im Körper GF(2^{128})
- $\oplus$ Bitwise XOR
- $\otimes$ Modulare Multiplikation mit Binärkoeffizienten modulo $x^{128} + x^7 + x^2 + x + 1$.

Übung

3.1. Der Initialisierungsvektor (IV) bei CBC

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

3.2. Padding

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

3.3. Auswirkungen eines Bitflips

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

3.4. Nonce bei OFB

Warum muss der IV bei OFB eine Nonce sein?

Eine Nonce (🇺🇸 *Number used ONCE*) ist eine Zahl, die nur einmal für die Ausführung des Verschlüsselungsalgorithmus verwendet wird.

Übung

3.5. CTR Mode - Anforderungen and den IV?

Wenn beim Counter Mode garantiert ist, dass der Schlüssel niemals wiederverwendet wird (zum Beispiel, weil er ein Session Key ist), kann dann für die Nonce *in diesem speziellen Fall* auch eine Konstante verwendet werden?

3.6. ECB identifizieren

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

3.7. XTS-AES

Wie viele Blöcke hat eine Dateneinheit, wenn ein Festplattensektor 4 KiB groß ist?

Welchen praktischen Vorteil hat es, das der Hash T vor und nach der Verschlüsselung des Klartextes mit dem aktuellen Wert XOR-verknüpft wird?

Übung

3.8. OFB-Modus

Verwenden Sie den OFB-Modus in Kombination mit einer Caesar-Chiffre, bei der die *Blockgröße* ein Zeichen sei.

Der Schlüssel ist die Anzahl der Zeichen, um die Sie ein Zeichen verschieben wollen - wie zuvor. Der IV ist ein Zeichen. Damit Sie ein XOR durchführen können, ordnen wir jedem Zeichen einen Wert zu und erweitern das Alphabet um die folgenden 6 Zeichen: { 1, 2, 3, !, ?, _ }. Auf diese Weise ist es immer möglich, ein sinnvolles Zeichen auszugeben.

Daraus ergibt sich die folgende Kodierung:

Idx.	Zeichen	Binär
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

Idx.	Zeichen	Binär
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

Idx.	Zeichen	Binär
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 die Nachricht "hello" (k = 3 und IV = !) mit dieser Chiffre.
- Entschlüsseln Sie nun die Nachricht "OCEBL_RLI1MELOA". Der IV ist A und der Schlüssel ist 3.
- Welchen Effekt hat die Anwendung des OFB-Modus auf die Nachrichten?