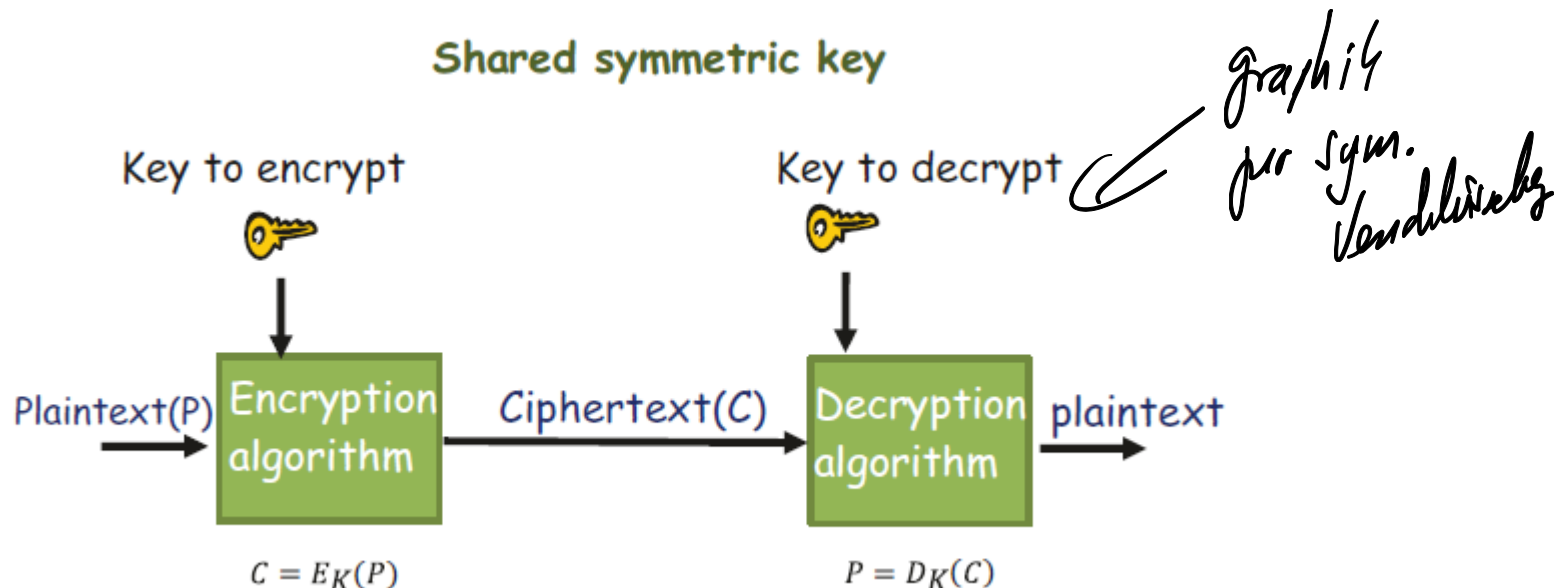


Kryptologie ICS.KRYPTO

Folien zur Präsenz 2 «Symmetrische Kryptographie», FS 24, V4.2



Graphik aus «New Frontiers in Cryptography», Khaled Salah Mohamed, Springer Verlag

©Josef Schuler, dipl. math., dipl. Ing. NDS ETHZ, MSc Applied IT-Security, Feldhof 25, 6300 Zug, j.schuler@bluewin.ch resp. josef.schuler@hslu.ch

Inhaltsübersicht

- Eine Einführung in die symmetrische Kryptographie
 - Diverse Funktionsprinzipien (Stromchiffre versus Blockchiffre) = Kap. 7.1 & 7.2
 - Diverse Funktionstechniken und –Modi von Blockchiffren = Kap. 8
- Mathematische Grundlagen
 - Binomialkoeffizienten
 - XOR-Operation (von HEX-Zeichen)

Lernziele

- Ich habe ein gutes Verständnis der Stromchiffren und Blockchiffren erlangt.
- Ich kenne die verschiedenen Modi der Blockchiffren und kann deren Stärken und Schwächen benennen.
- Ich kann eine krypt. Zeichnung in math. Form überführen und umgekehrt.
- Ich kenne den Begriff der perfekten Sicherheit.
- Ich habe die Zusammenfassung zur Vorbereitung (= Präsenz 0) und den Inhalt der ersten zwei Präsenzen (= Präsenz 1 & 2) bearbeitet (z.B. die Basisaufgaben und die weiteren Aufgaben in den Folien und im Skript gelöst) und die – erweiterten – Grundlagen der Kryptographie verstanden.

Die Slogans zu dieser Präsenz

Zu Stromchiffren SC: Eine SC bietet keine Integrität!

Zu Blockchiffren BC: BC sind das «work horse» der Sicherheit!

Verweise zur Literatur

- **Theorie und (weitere Aufgaben):**

- JS Skripte „Einführung in die Kryptologie“, Kap. 7.1, 7.2 & 8. Wobei das Kap. 7.1.7 Nachtrag: Feistel-Chiffre **nicht Prüfungsstoff** ist.
- JS Skripte „Einf. in die Krypt.“, Kap. 28 „Perfekte Sicherheit“ (Begriff kennen, die Details und Aufgaben sind **nicht Prüfungsstoff**).
- In [CP-D] die Kap. 2.1, 2.2 zu den Stromchiffren und Kap. 5 zu den Blockchiffren.
- Die Kapitelnummerierung in den folgenden Folien entspricht derjenigen im oben erwähnten JS Skript „Einführung in die Kryptologie“. D.h. die Details zu den Folien können im Skript nachgelesen werden. Zudem hat es im Skript weitere Übungen und Beispiele. **Die Aufgabennummerierung im JS Skript «Einführung in die Kryptologie» und in den vorliegenden Folien stimmen nicht überein!**
- **Wichtig:** Es ist unbedingt zu beachten, dass nur das Bearbeiten und Lernen der Folien nicht genügt. Das Durcharbeiten der oben erwähnten Kapitel in JS Skripte „Einführung in die Kryptologie“ sind absolut zentral zum Bestehen der Modulendprüfung.

- **Die Quellenangaben sind im JS Skript, nicht aber in den Folien enthalten!**

Kap. 7

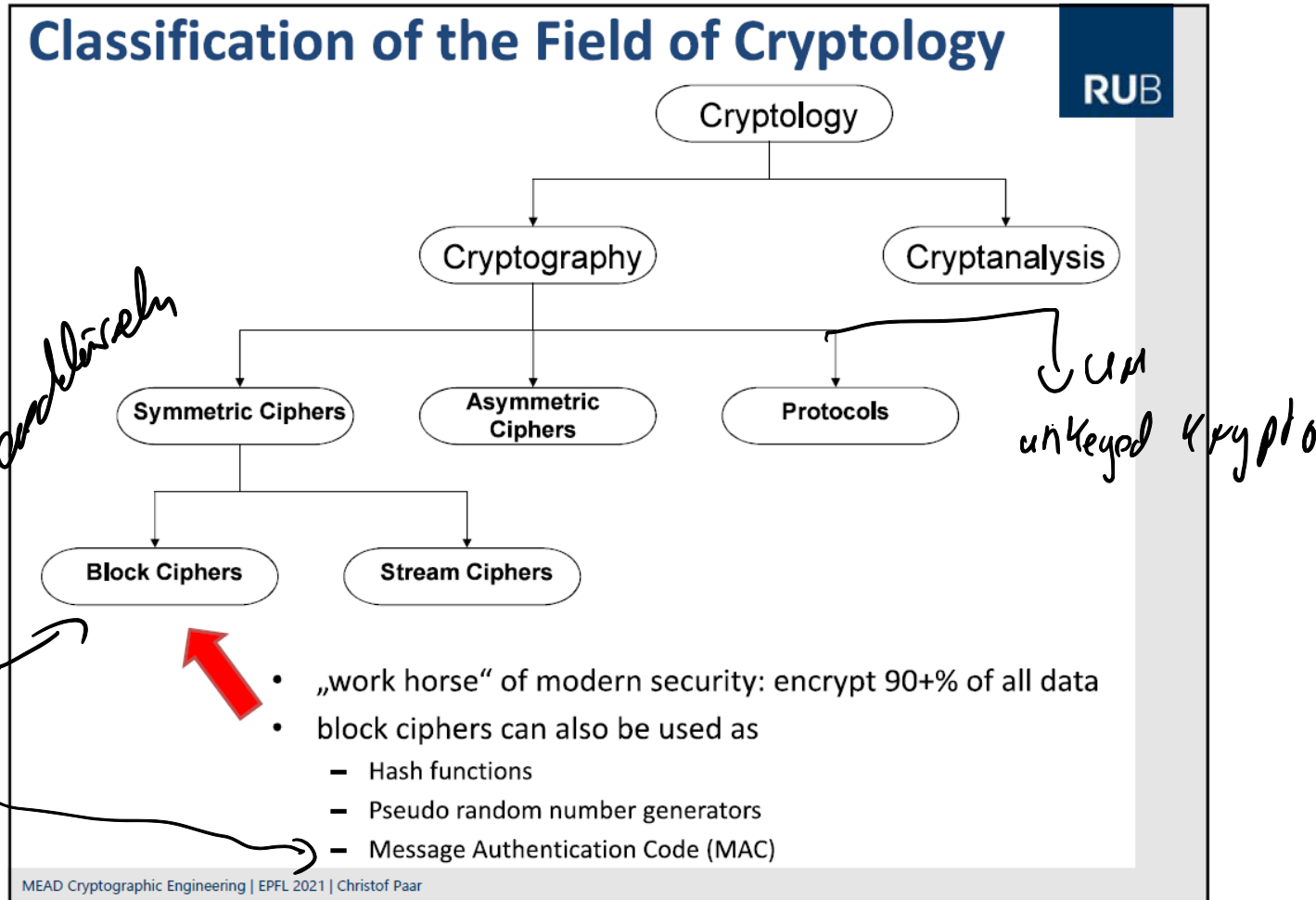
FUNKTIONSPRINZIPIEN VON ALGORITHMEN

Kap. 7.1

BESCHREIBUNG VON BLOCKCHIFFREN

Blockchiffre das «work horse» der Sicherheit

C. Paar zeigt im folgenden Bild sehr schön die Bedeutung der Blockchiffren.



Wichtige Ergänzung:

Blockchiffren sind deutlich resistenter gegenüber Quantencomputer als asym. Verfahren. Man entwickelt z.Z. „nur“ Post Quantum Algorithmen für asymmetrische Verfahren.

Beschreibung von Blockchiffren

Übliche Blockgrößen:

$n = 64, 128, 256$ Bit

Anzahl Runden:

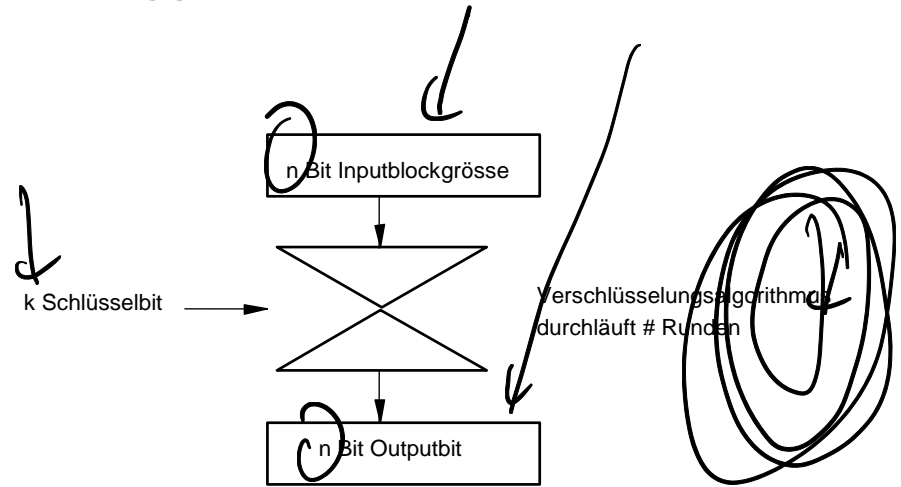
$\# = 10, 12, 14, 16$

Übliche Schlüssellängen

$k = 56$ (DES), 112 (3DES_2key),

$k = 128$ (AES), 168 (3DES_3key),

$k = 192, 256$ (AES) Bit



Charakteristika:

- Symmetrische Verfahren
- Blockgröße, Schlüssellängen und Rundenanzahl (= Teilschritte) sind die Kerngrößen
- Meistens in HW und in SW erhältlich
- Als Verschlüssler wie als Authentizierer verwendbar
- Verschiedene Betriebsarten (siehe Kap. 8)

• Einsatz:

- Bankenwelt
- Internet (SSL, TLS, IPSec, PGP)
- ePassport
- Message Dienste (z.B. WhatsApp oder Telegramm) → **Achtung:** in anderen Diensten werden Stromchiffrierer verwendet.

Produkte von Blockchiffren

Produkt	Blockgrösse	Schlüssellänge	# Runden	Bemerkungen
DES	64	56	16	<ul style="list-style-type: none"> Darf nur noch als Triple DES verwendet werden.
3DES mit 2 Schlüsseln	64	112	16	<ul style="list-style-type: none"> Ist – gemäss NIST – seit 2009 nicht mehr erlaubt. Sicherheit ist deutlich kleiner als 112 Bit (chosen plaintext attack), nämlich nur ca. 57 – 60 Bit. Also nicht mehr verwenden.
3DES mit 3 Schlüsseln (#)	64	168	16	<ul style="list-style-type: none"> Die effektive Schlüsselstärke ist – bez. einer meet in the middle attacke - aber <u>nur</u> 112 Bit. Ist – gemäss NIST – bis 2030 erlaubt, da bis dann 112 Bit genügen sollten.
IDEA	64	128	8	<ul style="list-style-type: none"> Blockverschlüssler in PGP. Schnell Hat sich wegen den Lizenzgebühren beim kommerziellen Einsatz nicht durchgesetzt.
SAFER-SK128	64	128	6 – 16	<ul style="list-style-type: none"> In HW und SW schnell Gratis Designed für 8- und 16-Bit Arithmetik (für Chipkarten)
RC-5 (*)	64	Bis 2048 128 (Default)	12	<ul style="list-style-type: none"> War Vorbild zum AES.
AES(****)	128 (**)	128, 192 oder 256	10, 12, 14 (***)	<ul style="list-style-type: none"> Nachfolger von DES Standardisiert
PRESENT	64	80/ 128	31	<ul style="list-style-type: none"> Als Lightweight Blockcipher standardisiert. Wird in div. Kleingeräten verwendet.
Andere (##)				<ul style="list-style-type: none"> Nur Standards benutzen.

Produkte von Blockchiffren, Bemerkungen

- (#) Der 3DES wird immer noch häufig eingesetzt (z.B. standardmässig bei ePassport).
Der Grund liegt darin, dass in einer Chipkarte mit DES-Chip der DES i.d.R. 2 mal schneller ist als ein AES.
- (*) Im Prinzip ist RC-5 voll parametrisierbar in (Blocklänge/Schlüssellänge/#Runden)
- (**) 128 Bit Blockgrösse ist im Standard so festgelegt. Der Algorithmus (Rijndael = Abkürzung der Namen der zwei Kryptologen Vincent Rijmne und Joan Daemen) liesse aber jedes Vielfache von 32 zu [GS].
- (***) Abhängig von der Schlüsselgrösse [GS].
- (****) Es ist zwischen dem Standard AES u. dem Algorithmus Rijndael zu unterscheiden.
- (##) Viele weitere Algorithmen, siehe z.B. bei der Aufzählung der AES Finalisten. Es gibt aber noch unzählige weitere.

Schlussbemerkung = sinngemässe Wiedergabe aus [RO1]:

Es gibt (in den meisten Fällen) keinen Grund etwas anderes als den AES zu nehmen, so lange dieser als sicher gilt. Z.Z. gibt es keine Hinweise, dass AES in nächster Zeit unsicher würde. AES dominiert das Feld der E2EE (End-to-End Encrypted Message Dienste wie WhatsApp u.v.m.).

Das deckt sich auch mit der [BSI1] Empfehlung.

Eine Ausnahme bilden Kleingeräte, für die PRESENT als Lightweight Blockcipher standardisiert ist.

ISO-18033 – Encryption Algorithms – Block Ciphers

This part of [ISO/IEC 18033](#) specifies block ciphers. A block cipher maps blocks of n bits to blocks of n bits, under the control of a key of k bits. A total of seven different block ciphers are defined. They are categorized in [Table 1](#).

Table 1 — Block ciphers specified

Block length	Algorithm name	Key length
64 bits	TDEA (4.2)	128 or 192 bits (@)
	MISTY1 (4.3)	128 bits
	CAST-128 (4.4)	
	HIGHT (4.5)	
128 bits	AES (5.2) (#)	128, 192 or 256 bits
	Camellia (5.3)	
	SEED (5.4)	128 bits

(@) Beim TDEA (Tripple-DES mit 2 resp. 3 Schlüsseln) ist die wirkliche Schlüsselgrösse natürlich nur 112, resp. 168 Bit. Die Angabe im Standard ist zumindest verwirrend, um nicht einfach „falsch“ schreiben zu müssen. Der DES hat nur eine effektive Schlüssellänge von 56 Bit, obwohl 64 Bit eingegeben werden. Jedes 8-te Bit ist ein Parity-Bit → cf. nachfolgende Folien.

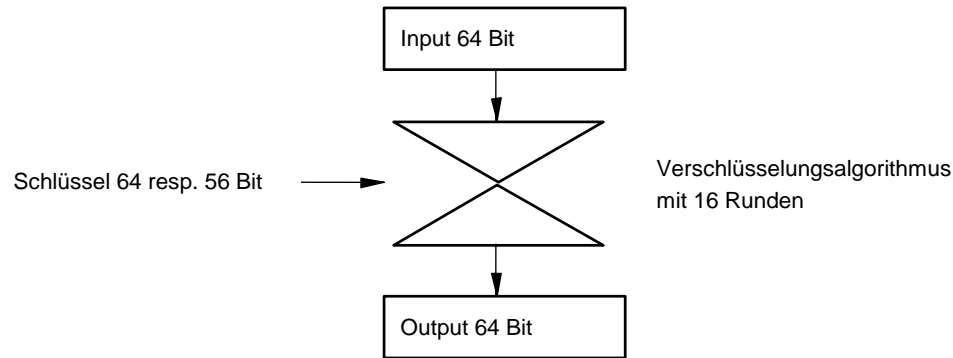
(#) Sowohl das BSI in [BSI1] wie auch Rolf Oppliger in [RO1] empfehlen den AES zu verwenden.

Allgemeine Eigenschaften von Blockchiffren

1. Beim Ändern eines Inputbits, wird bei gleichem Schlüssel – im statistischen Mittel – die Hälfte der Outputbits geändert.
2. Beim Ändern eines Schlüsselbits, wird bei gleichem Input – wiederum im statistischen Mittel – die Hälfte der Outputbits geändert.
3. In der Fachliteratur werden diese Effekte im Zusammenhang mit den Begriffen „Diffusion“, „Konfusion“ und „Avalanche-Effekt“ resp. „Lawineneffekt“ im Detail erörtert. → Wir gehen nicht weiter auf diese Begriffe ein, cf. Lit. wie [CP-D], [BS].
4. Mit Hilfe von sogenannten Binomialkoeffizienten kann man zeigen, dass es am idealsten ist, wenn die Hälfte der Outputbits geändert wird. Denn die Anzahl der Möglichkeiten sind maximal, resp. die Wahrscheinlichkeit zum Erraten minimal.
5. Allgemein ist $\binom{n}{m} = \frac{n!}{m! \cdot (n-m)!}$ bei $m = n/2$ maximal.
 - D.h. $\binom{n}{n/2} = \frac{n!}{(n/2)! \cdot (n-n/2)!} = \frac{n!}{(n/2)! \cdot (n/2)!} = \frac{n!}{[(n/2)!]^2}$ ist maximal.
 - Für $n = 64$ gibt das dann $\binom{64}{32} = \frac{64!}{32! \cdot (64-32)!} = \frac{64!}{32! \cdot 32!} = 1,8 \cdot 10^{18}$
 - Und bei $n = 128$: $\binom{128}{64} = \frac{128!}{64! \cdot (128-64)!} = \frac{128!}{64! \cdot 64!} = 2,4 \cdot 10^{37}$
 - Die Mitte ist maximal, cf. auch Pascal'sches Dreieck

$$\binom{64}{29} = \binom{64}{35} = 1,39 \cdot 10^{18}; \quad \binom{64}{30} = \binom{64}{34} = 1,62 \cdot 10^{18}; \quad \binom{64}{31} = \binom{64}{33} = 1,78 \cdot 10^{18}$$

Beispiel 1: DES (Data Encryption Standard)

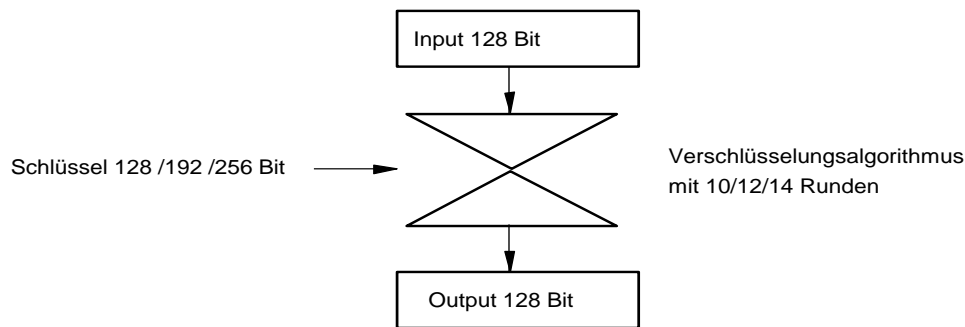


Es wird ein 64 Bit Schlüssel eingegeben, aber jedes 8-te Bit ist ein Parity-Bit, das je nach Algorithmus geprüft oder ignoriert wird. Wenn das Parity-Bit nicht geprüft wird, kann man sagen, dass es zu jedem Schlüssel weitere 255 „identische“ Schlüssel gibt. Alle 256 Schlüssel unterscheiden sich nur in den Parity-Bits. Somit beträgt die effektive Schlüsselgrösse nur 56 Bit.

Fakts zum DES (Data Encryption Standard), tw. aus [CP21] zitiert:

- Einer der am meisten verwendeten Blockchiffren in den 1980-er bis 2000-er Jahre.
- DES ist eine sogenannte Feistelchiffre (der Begriff wird in diesem Modul nicht weiter erörtert).
- **Vorteil:** „Encrypt“ & „Decrypt“ sind bis auf die Reihenfolge der Verwendung der Schlüsselbits gleich.
- 3DES ist gemäss NIST noch bis 2030 erlaubt und wird für epassport und banking verwendet.
- 1998 ist das US Patent abgelaufen.
- DES beeinflusste viele weitere Blockchiffrealgorithmen.
- Der Single-DES gilt als geknackt (2008: innerhalb von 6 Tagen mit 10'000 \$ Kosten)
- DES ist insbesondere wegen (unnötigen) Permutationen in HW viel schneller als in SW. Z.B.:
 - In (Spezial-)HW > 10 Gbit/sec.
 - In (hoch optimierter) SW ca. 400 Mbit/sec. Ansonsten in SW viel, viel langsamer.

Beispiel 2: AES (Advanced Encryption Standard)



AES wurde in einem Wettbewerb der NIST (US National Institute of Standards and Technology) ausgewählt. Es war ziemlich überraschend, dass sich der einzige nicht US-Algorithmus durchgesetzt hat.

Weitere Fakten zum AES (Advanced Encryption Standard), tw. aus [CP21] zitiert:

- Seit 2000 als Nachfolger vom DES standardisiert, er wird (voraussichtlich) die nächsten 20 Jahre Bestand haben.
- Der AES ist nicht nur sicherer als der DES, sondern in HW wie als SW auch viel schneller.
- Der AES ist insbesondere auch von der NSA (National Security Agency) zugelassen, um TOP SECRET Dokumente zu verschlüsseln.
- Der AES ist keine Feistelchiffre – der Begriff ist im JS Skript erörtert, ist aber nicht Prüfungstoff. **Nachteil:** „Encrypt“ und „Decrypt“ sind unterschiedlich, sie brauchen unterschiedliche Algorithmen.
- Der AES braucht für „Decrypt“ **doppelt** so lange wie „Encrypt“.
- Der AES mit 256 Schlüsselbits braucht nur ca. 40% länger als AES-128, dies weil 14 statt 10 Runden.
- Z.Z. sind keine relevanten Attacken bekannt, die besser als Brute-Force Attacken sind. Es gibt allerdings einen (theoretischen) Angriff, so dass der AES mit 256 Bit Schlüsselgröße „nur“ so sicher ist, wie wenn er die Schlüsselgröße 224 Bit hätte.
- Der AES ist „Software“-freundlich; da lernte die NIST offensichtlich etwas aus den Vorwürfen zum DES.
- Aber unterschiedliche Implementationstechniken bei Verwendung von 8-Bit Prozessoren (z.B. Chipkarten) oder 32-Bit Prozessoren (z.B. HSM). Eine SW, die für einen 32-Bit Prozessor geschrieben wurde, würde prinzipiell auf einem 8-Bit Prozessor laufen, wäre aber sehr langsam.

Einschub: Lightweight Cryptography

Warum Lightweight Cryptography?

Im Rahmen von „Internet of Things“ (IoT), Sicherung von „Gebrauchsgeständen“ wie Autos usw., Kommunikation mittels RFID und Einsatz von Low-Energy Devices ist der Wunsch gross, einen Blockchiffrierer zu haben, der eine gute Performance und einen kleinen Stromverbrauch hat und zudem wenig Platz braucht.

Lightweight Cryptography

RUB

Goals of lightweight crypto

1. Low power
2. Low energy
3. Low monetary costs
4. ... with acceptable performance

HW vs. SW

Dedicated, optimized hardware is much preferable over software (e.g., on 8 bit CPU) since it optimizes 1.-4. better

Why?

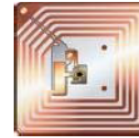
Many applications (see next slide)...

Einschub: Lightweight Cryptography

Applications for Lightweight Crypto

RUB

1. “We need RFID security with less than 2000 gates”
Sanjay Sarma, AUTO-ID Labs, CHES 2002



2. Securing sensor networks, e.g., infrastructure



3. Anti-Counterfeiting (\$3 trillions damage due to product piracy*)



*Source: www.bascap.com

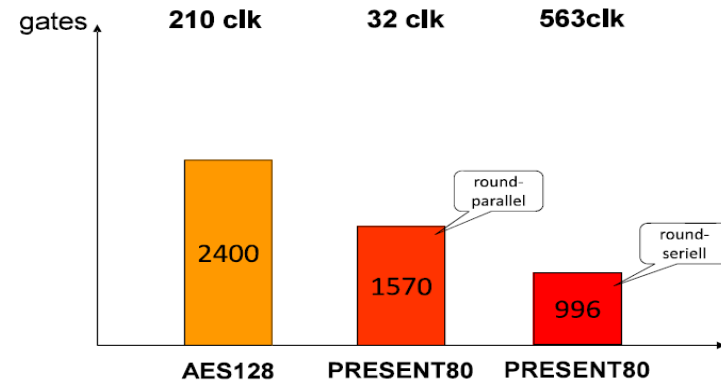
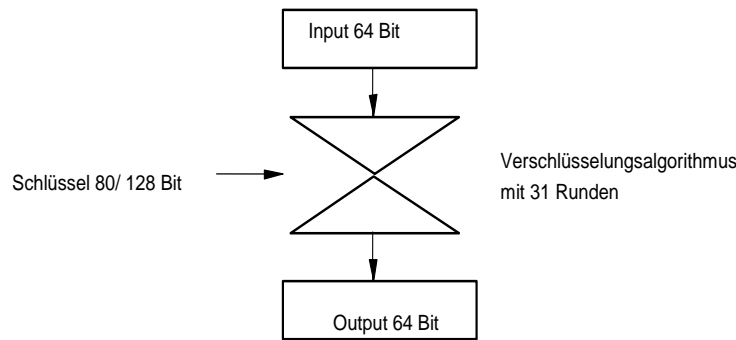
⇒ Mainly authentication & identification: can both be fixed with cryptography

MEAD Cryptographic Engineering | EPFL 2021 | Christof Paar

Bemerkung:

Mit «trillions» ist das amerikanische trillion = 1000 billion = 1000 Milliarden = 10^{12} gemeint!!
Im British english und deutsch ist eine Trillion = 1000 Billiarden = 1'000'000 Billionen = 10^{18} .
Die oben erwähnte Zahl von 3 trillions \$ entspricht in etwa der Wirtschaftsleistung von Italien.

Beispiel 3: PRESENT



- Time-area product 10 times better than smallest AES (90% energy savings)
- 128 bit version adds appr. 250 gates

Weitere Fakts zum PRESENT, tw. aus [CP21] zitiert:

- Seit 2012 standardisiert.
- Gibt es als 80 & 128 Bit Algorithmus.
- Braucht wenig Energie in HW und SW.
- SW i.d.R. für 8-Bit Prozessor optimiert.
- In optimierter HW bis 100-mal geringerer Stromverbrauch als in SW.
- Die HW für den 80 Bit Algorithmus braucht in der seriellen Implementation nur ca. 1000 Gatter. Damit kommt er nahe an die theoretische minimale Anzahl (800) von Gattern aus.
- Für den 128 Bit Algorithmus braucht es nur ca. 250 Gatter mehr.
- Es gibt viele Vorschläge für weitere ähnliche Algorithmen.
- Es gibt aktuell über 20 Kryptoanalyse Paper zu PRESENT, z.Z. sind keine relevanten Attacken bekannt, die besser als Brute-Force Attacken sind.
- Die HW für den 80 Bit Algorithmus in der parallelen Implementation braucht zwar mehr Gatter als die SW, aber die Kennzahl "Zeit · Platz" = $\text{clk} \cdot \text{Gatter}$ ist ca. 10-mal kleiner als bei AES 128 (siehe Diagramm oben).

	AES	PRESENT 80, parallel	PRESENT 80, seriell
"Zeit · Platz"	$210 \cdot 2400 = 504'000$	$32 \cdot 1570 = 50'240$	$563 \cdot 996 = 560'748$

Basis-Test Präsenz 2 zu Kap. 7.1

16.6.15

Aussage	Richtig oder falsch?	Begründung
Als Blockchiffre ist AES mit 256 Bit Schlüssellänge zu empfehlen.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	inkl. nächste Folie
Ein Tripple-DES mit 3 versch. Schlüsseln hat eine Sicherheit von 112 Bit.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	mit Beispiel 9
Wird im Input einer Blockchiffre ein Bit geändert, dann ändert – bei gleichem Schlüssel – die erste Hälfte der Bits des Chiffrats.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	und Aufgabe 7.0
Wird ein Schlüsselbit einer Blockchiffre ein Bit geändert, dann ändern sich – bei gleichem Input – im stat. Mittel die Hälfte der Bits des Chiffrats.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Das Verschlüsseln und Entschlüsseln dauern beim DES gleich lang.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Das Verschlüsseln und Entschlüsseln dauern beim AES gleich lang.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Beim AES-256 dauert die Verschlüsselung etwa doppelt so lange wie beim AES-128.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Lightweight Cryptography wird in Zukunft immer wichtiger.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Es gibt noch keine Lightweight Crypto-Algorithmen.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	

Einschub Kap. 28

ANHANG 6

PERFEKTE SICHERHEIT (*)

(*) Nur der Inhalt der Folien ist prüfungsrelevant!



Perfekte = informationstheoretische Sicherheit

Perfekte = informationstheoretische Sicherheit ...

- ... meint, dass man – selbst mit unendlich vielen Ressourcen – nicht besser ist als mit raten.
- ... meint, dass ein Chiffre keine Informationen über den Klartext liefert. Das kann mathematisch wie folgt beschrieben, resp. definiert werden:
Es sei \mathcal{M} die Menge aller möglichen Klartexte und \mathcal{C} die Menge aller möglichen Chiffre. Perfekte Sicherheit gilt, wenn $\forall m \in \mathcal{M}$ und $\forall c \in \mathcal{C}$ gilt, dass
$$p(m|c) = p(m).$$
- ... liefern insbesondere ganz einfache Operationen wie XOR, Addition mod N, Multiplikation mod p usw.
- ... hat als Vorbedingung, dass der gewählte Schlüssel absolut zufällig ist.
- ... liefern insbesondere ganz einfache Operationen wie XOR, Addition mod N, Multiplikation mod p usw.

Beispiel 4:

Betrachten wir die folgende Situation. Wir verschlüsseln jeweils nur einen Buchstaben mit der Caesar-Chiffre und wählen jedes Mal den Schlüssel $k \in K = \mathbb{Z}_{26}$ zufällig und mit gleicher Wahrscheinlichkeit.

Dann ist diese Verschlüsselung eines Buchstabens perfekt sicher, d.h. selbst mit unendlich vielen Ressourcen ist man mit allen möglichen Analysen nicht erfolgreicher als mit raten.

Aufgabe 7.0 Da muss es ja wohl einen Haken geben, oder? Klären Sie auf!

Kap. 7.2

BESCHREIBUNG VON STROMCHIFFREN

12.03.16



Beschreibung von Stromchiffren, XOR

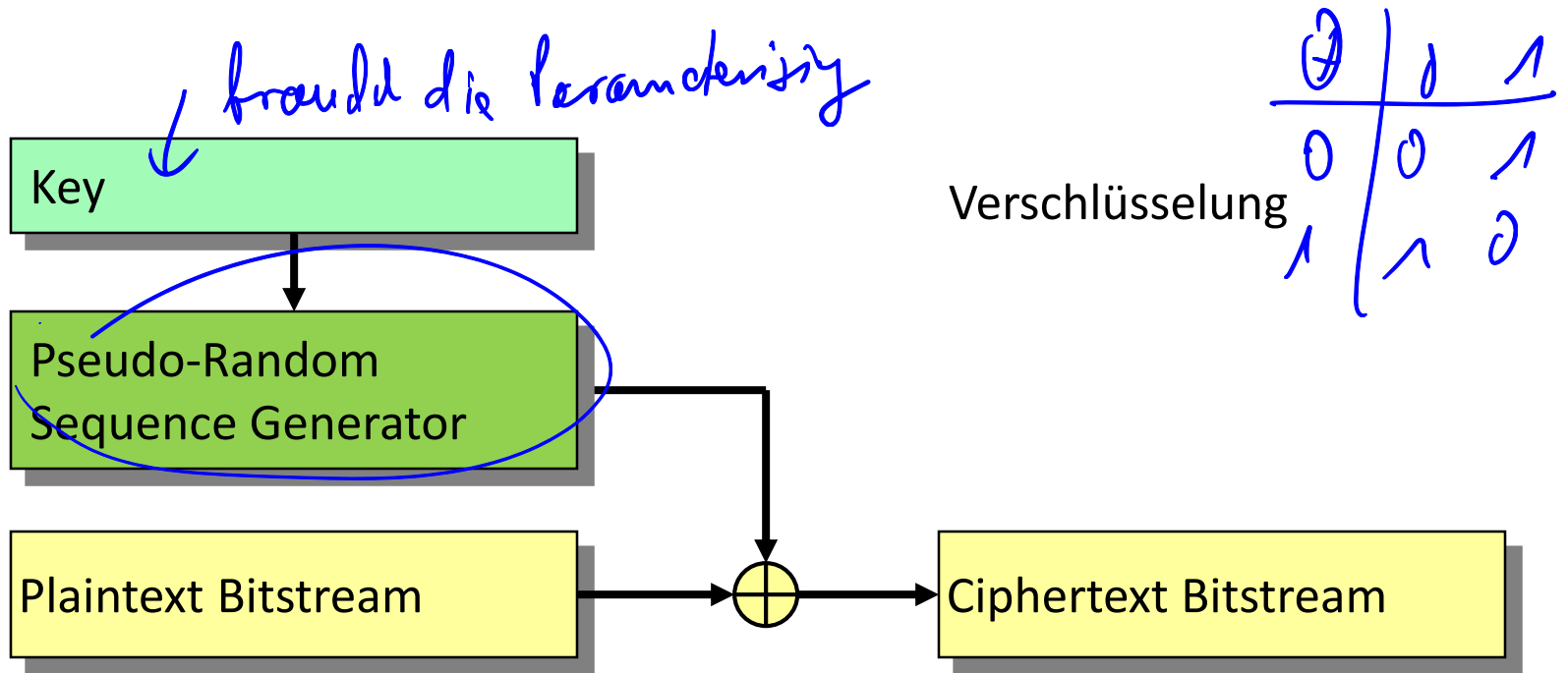
\oplus	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B
5	5	4	7	6	1	0	3	2	D	C	F	E	9	8	B	A
6	6	7	4	5	2	3	0	1	E	F	C	D	A	B	8	9
7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	8	B	A	D	C	F	E	1	0	3	2	5	4	7	6
A	A	B	8	9	E	F	C	D	2	3	0	1	6	7	4	5
B	B	A	9	8	F	E	D	C	3	2	1	0	7	6	5	4
C	C	D	E	F	8	9	A	B	4	5	6	7	0	1	2	3
D	D	C	F	E	9	8	B	A	5	4	7	6	1	0	3	2
E	E	F	C	D	A	B	8	9	6	7	4	5	2	3	0	1
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Aufgabe 7.1 Berechnen Sie nun $B \oplus A$: $= C$

Aufgabe 7.2 Was für ein Resultat gibt es, falls man zwei gleiche Ausdrücke miteinander „XORed“?

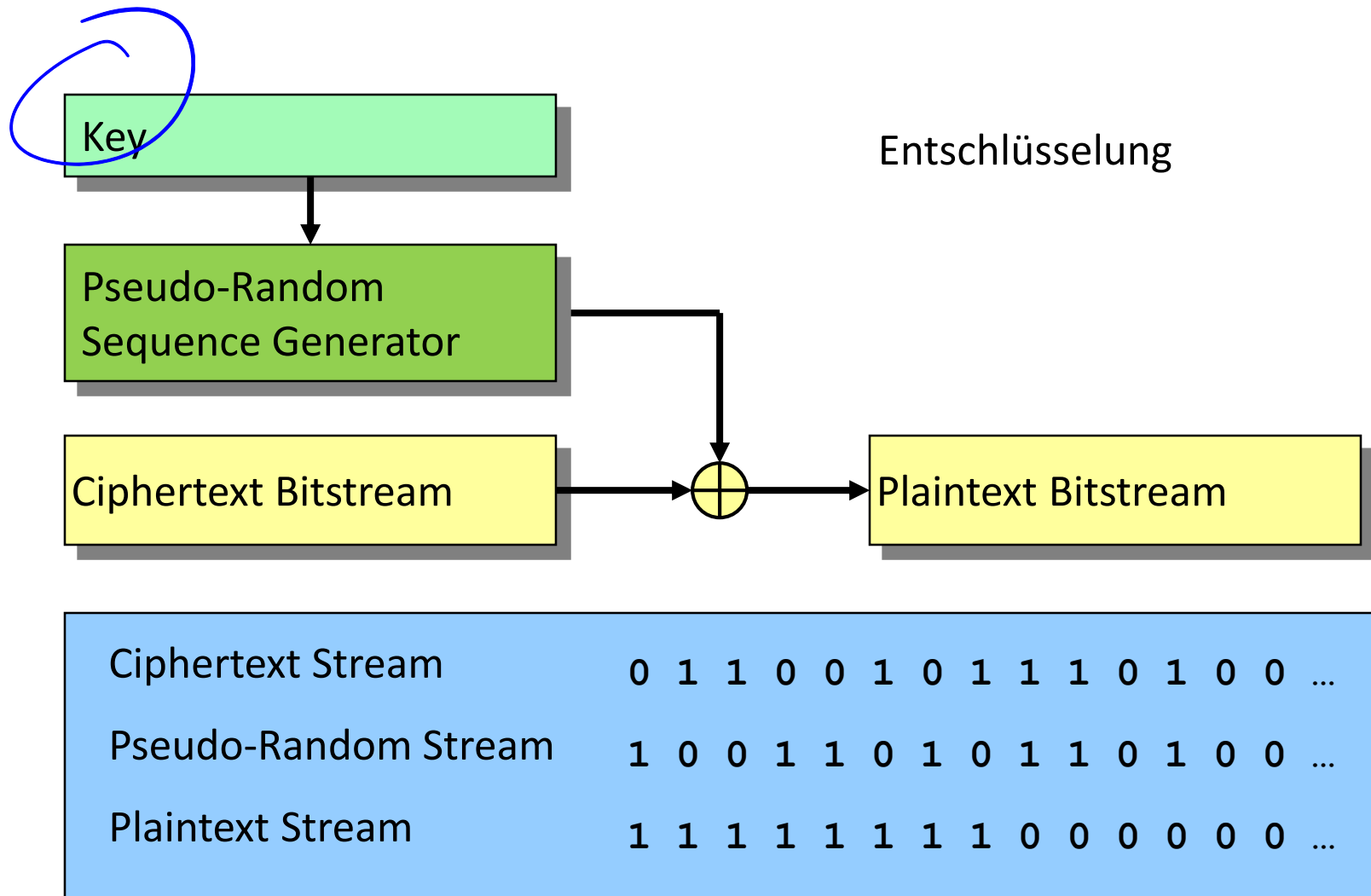
$= 0$

Beschreibung von Stromchiffren, encryption



Plaintext Stream	1	1	1	1	1	1	1	1	0	0	0	0	0	0	...
Pseudo-Random Stream	1	0	0	1	1	0	1	0	1	1	0	1	0	0	...
Ciphertext Stream	0	1	1	0	0	1	0	1	1	1	0	1	0	0	...

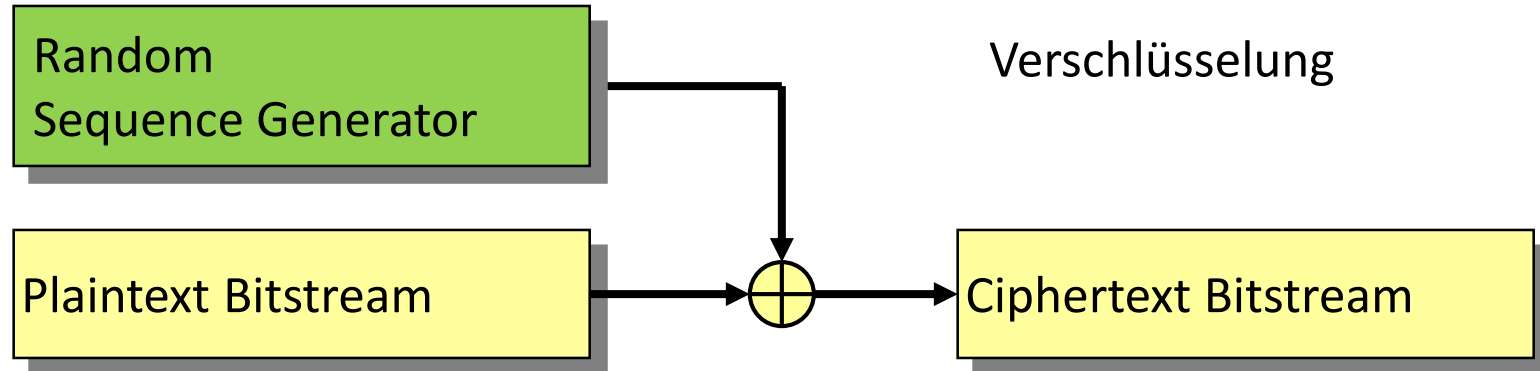
Beschreibung von Stromchiffren, decryption



[?] **Stromchiffren \cong One-Time-Pad (OTP)**

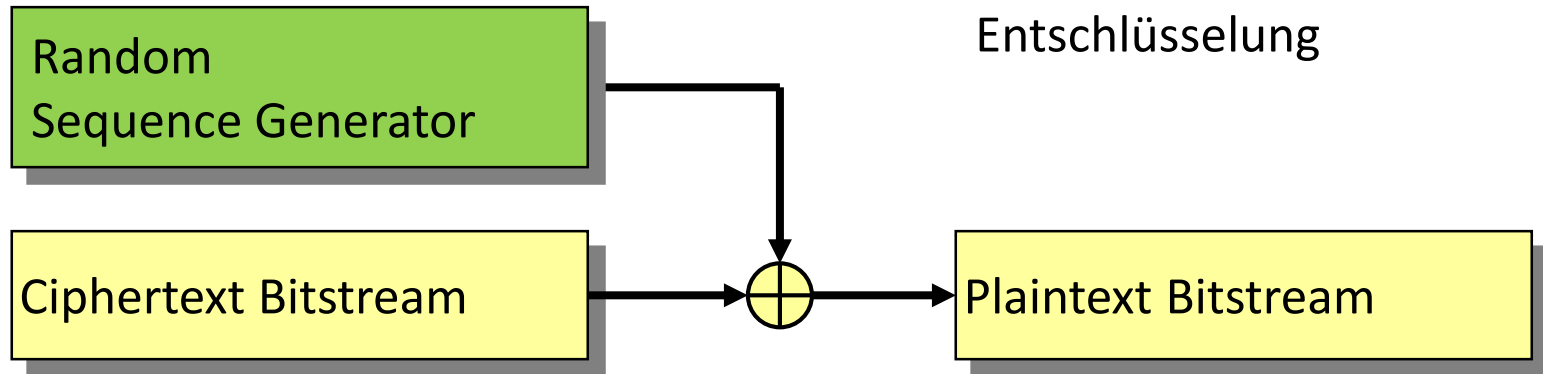
Der One-Time-Pad (OTP) ist eine Stromchiffre mit einem echt zufälligen (und nicht nur pseudo-zufälligen) Schlüsselgenerator.

Das Problem ist nur, wie transportiert man dann diesen echt zufälligen Schlüssel zum Empfänger. Und zudem muss die Schlüssellänge so gross wie der Klartext sein. Dafür ist OTP perfekt sicher (Details, cf. Kap. 28, „Perfekte Sicherheit“ im JS Skript, wobei nur der vorherige Einschub, aber nicht das ganze Kap. 28 prüfungsrelevant sind).



Plaintext Stream	1	1	1	1	1	1	1	1	0	0	0	0	0	0	...
Random Stream	1	0	0	1	1	0	1	0	1	1	0	1	0	0	...
Ciphertext Stream	0	1	1	0	0	1	0	1	1	1	0	1	0	0	...

Stromchiffren $\Leftarrow \Rightarrow$ *One-Time-Pad (OTP)*



Ciphertext Stream	0	1	1	0	0	1	0	1	1	1	0	1	0	0	...
Random Stream	1	0	0	1	1	0	1	0	1	1	0	1	0	0	...
Plaintext Stream	1	1	1	1	1	1	1	1	0	0	0	0	0	0	...

Beschreibung von Stromchiffren, Diverses

Verschlüsselung: $C = M \oplus S$

Entschlüsselung: $C \oplus S = (M \oplus S) \oplus S = M \oplus (S \oplus S) = M \oplus 0 = M$

Charakteristika/Produkte:

Es gibt „nur“ eine Stromchiffre, das XOR; aber die Erzeugung der Pseudo Random Bits, also der Pseudo Random Generator – unten Keygenerator genannt – ist unterschiedlich.

- Der Keygenerator ist ...
 - ... in gewissen Anwendungen eine Kombination von Schieberegistern → out of scope.
 - ... oft (Bankenwelt) ein Blockverschlüssler in einem besonderen Modus (cf. Kap. 8.3.).
 - ... oft (Netzwerkwelt) eine spezielle Konstruktion mit Hashfunktionen wie SHA-1.
- Stromchiffren – engl. Stream cipher – sind symmetrische Verfahren
- Vor allem bei Link-Verschlüssler als HW erhältlich.
- Nur als Verschlüssler brauchbar, d.h. ein Stromchiffrierer bietet **keine Integrität/Authentizität**. → **siehe auch Beispiel im Skript Kap. 7.2.3!**
- Ein Verändern im Chiffretext wirkt sich nur auf die direkt betroffenen Zeichen aus.
- Wird aber ein Bit aus dem Chiffretext entfernt oder zugefügt, so gerät beim Entschlüsseln – ausser es handelt sich um selbstsynchronisierende Stromchiffrierer – alles ausser „Rand und Band“.

Beschreibung von Stromchiffren, Fort.

Einsatz/Produkte:

1. Link encryption
2. A5 = Verschlüsselung vom Handy zur Funkantenne (A5 ein sich selbstsynchronisierender Stromchiffrierer; Grundlagen von U. Maurer, ETHZ)
3. RC-4 in WLAN Verschlüsselung (WEP), auch die 104 Bit Variante ist schlecht implementiert → besser WPA2 (Einsatz des Blockchiffrier-Algorithmus AES).
4. Neuere Produkte (z.B. Salsa20 oder ChaCha20) kommen in End-to-End Encrypted Message Diensten wie Threema (Salsa20) oder Signal resp. in deren Implementationen wie Wire (ChaCha20) vor.

Und es sei nochmals erwähnt:

Ein Stromchiffre bietet **keine Integrität/Authentizität → siehe Beispiel im Skript Kap. 7.2.3!**

Hausaufgabe

Bearbeiten Sie nun das oben erwähnte Beispiel in Kap. 7.2.3 im Skript „Einführung in die Kryptologie“.

Basis-Test Präsenz 2 zu Kap. 7.2

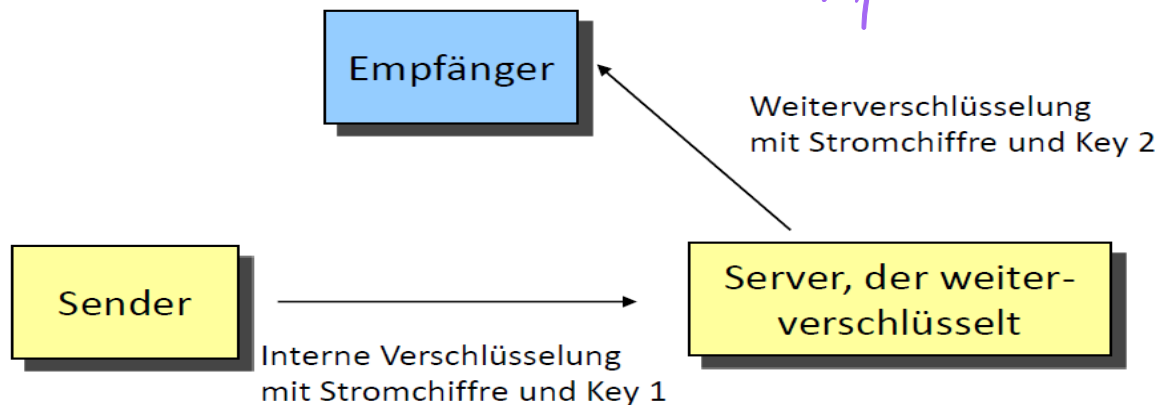
Aussage	Richtig oder falsch?	Begründung
Das wirkliche „Arbeitspferd“ der Kryptologie sind die Stromchiffren.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Bei Stromchiffren ist die Verschlüsselung und die Entschlüsselung exakt die gleiche Operation.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Es gibt mehrere Stromchiffre Verschlüsselungsalgorithmen.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Stromchiffre und One-Time-Pad sind identische Begriffe.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Mit Stromchiffren kann ich neben dem Verschlüsseln auch die Integrität einer Meldung gewähren.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
$12 \oplus 36 = 48$	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	

$$\begin{array}{r}
 12 \quad 0001 \ 0010 \\
 36 \quad 0011 \ 0110 \\
 \hline
 0010 \mid 0100 \\
 2 \quad 4
 \end{array}$$

16 4 40 → Paare bis 16h50

Aufgabe 7.5

Sie haben nun das folgende Schema einer Weiterverschlüsselung vor sich. Der Sender will dem Empfänger eine verschlüsselte Meldung übermitteln. Zunächst verschlüsselt der Sender die Meldung mit dem Schlüssel Key 1 = 8E. Die Meldung kommt zunächst zu einem Kommunikationsserver, der die erhaltene Meldung mit dem Key 2 = 58 weiterverschlüsselt, ohne sie vorher zu entschlüsseln. Die Verschlüsselungen sind in beiden Fällen Stromchiffren. Alle Werte sind in HEX. Wenn ihr TR HEX nicht rechnen kann, so benutzen so doch die HEX-XOR Tabelle in Folie 42, resp. im Skript Kap. 7.2.1.





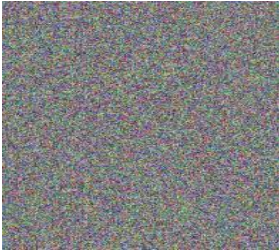

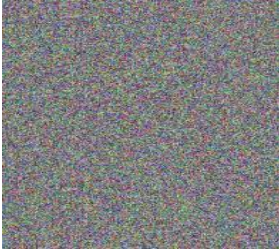
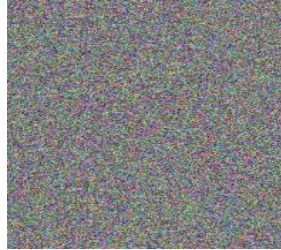
- a) Welchen Schlüssel Key 3 muss der Empfänger anwenden, wenn er die Entschlüsselung in einem Schritt durchführen will?
- b) Wie lautet die entschlüsselte Meldung, wenn der Empfänger die Meldung = 32 erhalten hat?

Aufgabe 7.6 Beurteilung einer Stromchiffre

- a) Welche(s) Schutzziel(e) müssen festgelegt sein, um den Einsatz einer Stromchiffre zu rechtfertigen?
- ☐ Schutz vor unberechtigtem Einfügen einer Meldung.
 - ☐ Schutz zum Entdecken, ob die Meldung verändert wurde.
 - ☐ Schutz um das Löschen einer Meldung zu entdecken.
 - ☐ Schutz, damit sich ein Gegner (Eve) nicht als Absender (Alice) ausgeben kann.
 - ☐ Schutz, damit sich ein Gegner (Eve) nicht als Empfänger (Bob) ausgeben kann.
 - ☐ Schutz gegen das Abhören der Meldung.
 - ☐ Schutz um zu Verhindern, dass der Sender behaupten kann, die Meldung nicht geschickt zu haben.
 - ☐ Schutz um zu Verhindern, dass der Empfänger behaupten kann, die Meldung nicht bekommen zu haben.
 - ☐ Keine der Angaben ist zutreffend.
- b) Mit was für Typen Gegner/Angriffen muss gerechnet werden müssen, um den Einsatz einer Stromchiffre zu rechtfertigen?
- ☐ Es muss hauptsächlich mit Insider Angriffen gerechnet werden.
 - ☐ Stromchiffre ist hauptsächlich ein Schutz gegen intelligente Gegner, wie z.B. Hacker, Geheimdienste usw., die einen passiven Angriff durchführen.
 - ☐ Stromchiffre ist hauptsächlich ein Schutz gegen intelligente Gegner, wie z.B. Hacker, Geheimdienste usw., die einen aktiven Angriff durchführen.
 - ☐ Nicht intelligente Gegner, wie z.B. ein Fehlverhalten (Übertragungsfehler) des Systems oder eines Menschen (Verschreiben).
 - ☐ Keine der Angaben ist zutreffend
- c) Bei der Stromchiffre handelt es sich um was für einen Mechanismus?
- ☐ Asymmetrische Verschlüsselung.
 - ☐ Schlüsselloser Kryptographischer Mechanismus
 - ☐ MAC-Berechnung
 - ☐ Authentizierprotokoll
 - ☐ Symmetrische Verschlüsselung
 - ☐ Digitale Signatur
 - ☐ Diffie-Hellman Schlüsselaustausch
 - ☐ Nicht kryptographische Hashfunktion (z.B. eine Prüffunktion)
 - ☐ Kryptographische Hashfunktion (z.B. SHA-1)

Aufgabe 7.7 Das Stromchiffre-Puzzle - Sender

Der Sender verschlüsselt zwei Nachrichten M_1 und M_2 mit einer Stromchiffre.

	$i = 1$	$i = 2$
Meldung M_i		
Key K_i		
Chiffre C_i $C_i = M_i \oplus K_i$		

Aufgabe 7.7 Der Empfänger und das Puzzle

Der Empfänger (das Kryptomodul) erhält zwei verschlüsselte Nachrichten, entschlüsselt sie und zeigt dem Betrachter nachher das folgende Bild:



- Was ging hier schief? Beim Sender? Beim Empfänger? Ev. bei beiden etwas? Hinterfragen Sie alle Möglichkeiten, die schief gehen konnten.
- Betrachten Sie die folgenden Puzzleteile (Sie brauchen nicht alle Teile!) **Tipp:** Schauen Sie an, was der Empfänger (möglicherweise) erhalten hat, und folgern Sie dann, was passiert sein könnte.

<ul style="list-style-type: none"> • $C_1 = M_1 \oplus K_1$ • $C_2 = M_2 \oplus K_2$ • $C_2 = M_2 \oplus K_1$ • $C_1 = M_1 \oplus K_2$ • $C_1 \oplus C_1 =$ • $C_1 \oplus C_2 =$ 	<ul style="list-style-type: none"> • $= (M_1 \oplus M_1) \oplus (M_2 \oplus M_2) =$ • $= (M_1 \oplus K_1) \oplus (M_2 \oplus K_2) =$ • $= (M_1 \oplus K_1) \oplus (M_2 \oplus K_1) =$ • $= (M_1 \oplus M_2) \oplus (M_2 \oplus K_1) =$ • $= (M_1 \oplus K_1) \oplus (M_2 \oplus M_2) =$ 	<ul style="list-style-type: none"> • $= M_1$ • $= M_2$ • $= M_1 \oplus M_1$ • $= M_1 \oplus M_2$ • $= M_2 \oplus M_2$
--	---	---

Aufgabe 7.7 Der Empfänger und das Puzzle

Der Empfänger (das Kryptomodul) erhält zwei verschlüsselte Nachrichten, entschlüsselt sie und zeigt dem Betrachter nachher das folgende Bild:



- Was ging hier schief? Beim Sender? Beim Empfänger? Ev. bei beiden etwas? Hinterfragen Sie alle Möglichkeiten, die schief gehen konnten.
- Betrachten Sie die folgenden Puzzleteile (Sie brauchen nicht alle Teile!) **Tipp:** Schauen Sie an, was der Empfänger (möglicherweise) erhalten hat, und folgern Sie dann, was passiert sein könnte.

- $C_1 = M_1 \oplus K_1$
- $C_2 = M_2 \oplus K_2$
- $C_2 = M_2 \oplus K_1$
- $C_1 = M_1 \oplus K_2$
- $C_1 \oplus C_1 =$
- $C_1 \oplus C_2 =$

- $= (M_1 \oplus M_1) \oplus (M_2 \oplus M_2) =$
- $= (M_1 \oplus K_1) \oplus (M_2 \oplus K_2) =$
- $= (M_1 \oplus K_1) \oplus (M_2 \oplus K_1) =$
- $= (M_1 \oplus M_2) \oplus (M_2 \oplus K_1) =$
- $= (M_1 \oplus K_1) \oplus (M_2 \oplus M_2) =$

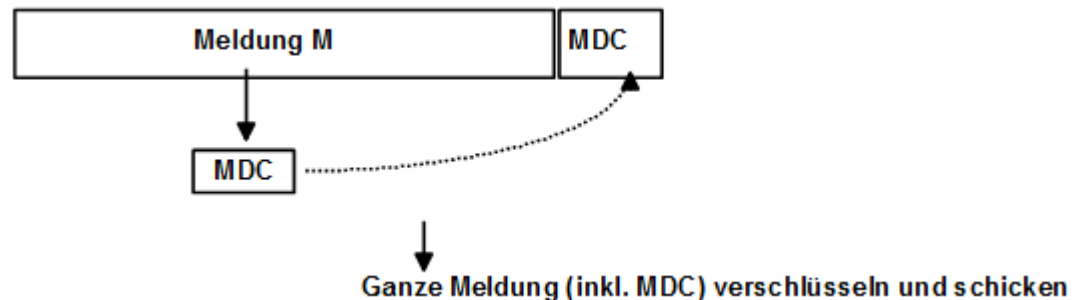
- $= M_1$
- $= M_2$
- $= M_1 \oplus M_1$
- $= M_1 \oplus M_2$
- $= M_2 \oplus M_2$

Kap. 8

TECHNIKEN BLOCKCHIFFREN

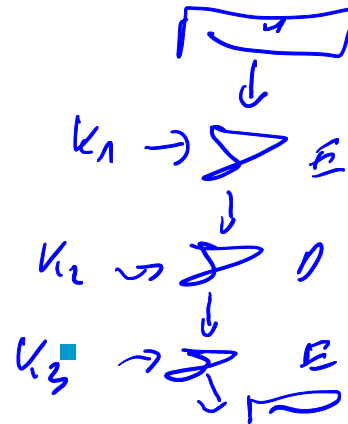
Einsatzmöglichkeiten von Blockchiffren

- Mit Blockchiffren kann man nicht nur – in diversen Betriebsmodi – Verschlüsseln. Nein, es gibt auch noch andere Einsatzmöglichkeiten für Blockchiffren.
 - Verschlüsselung mit Blockchiffren im Blockmodus, cf. Kap. 8.1.
 - CBC-MAC Berechnung cf. Kap. 8.2.
 - Schlüsselgenerierung für Streamcipher, d.h. Verschlüsselung mit Blockchiffren im Streamciphermodus cf. Kap. 8.3.
 - Randomgenerator z.B. für Sessionkeyerzeugung, cf. Kap. 8.4.
 - Das wird auch in der Übersichtszeichnung in Folie 6 unter dem roten Pfeil aufgelistet.
- Neben den im Folgenden behandelten Betriebsmodi gibt es noch eine ganze Reihe weiterer, z.B.
 - CFB
 - GCM...
- Spannend sind kombinierte Modi (Verschlüsseln und MAC'en) als Ersatz des Galois Counter Mode (GCM), da dieser Schwächen enthält. Mittels einem Wettbewerb wurden zwei Algorithmen eruiert: ACRON & AEGIS; wir gehen aber nicht weiter darauf ein.
- Die Idee von Kombi Modi ist nicht absolut neu, anbei der erste Typ.

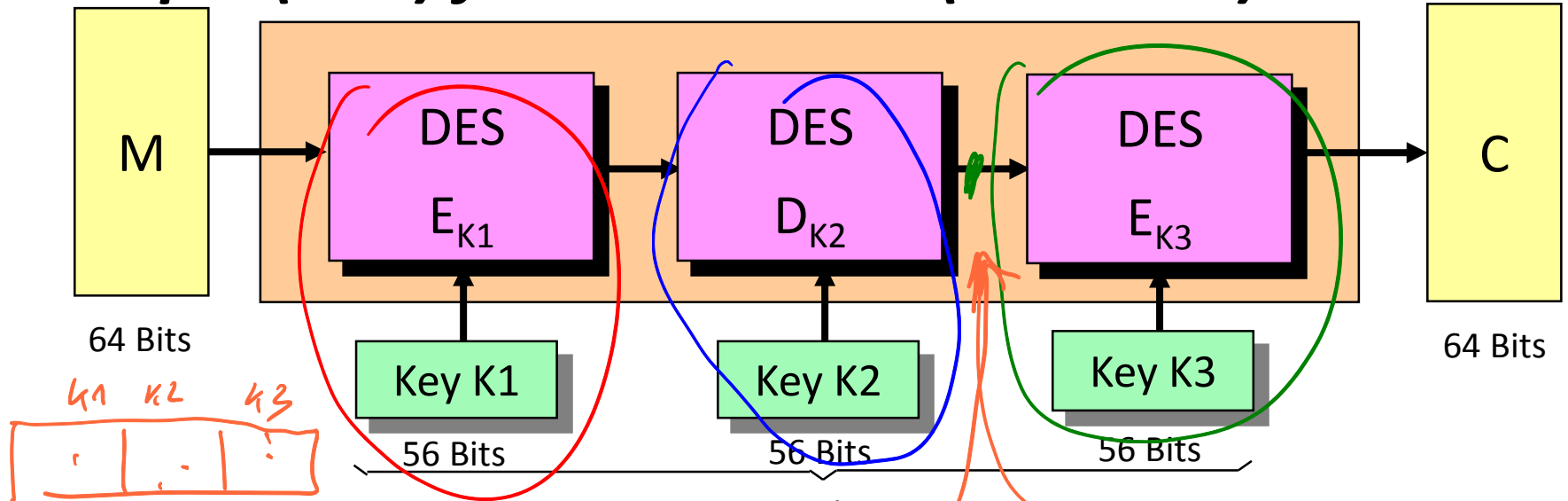


Kap. 8.1

VERSCHLÜSSELUNG MIT BLOCKCHIFFREN IM BLOCKMODUS

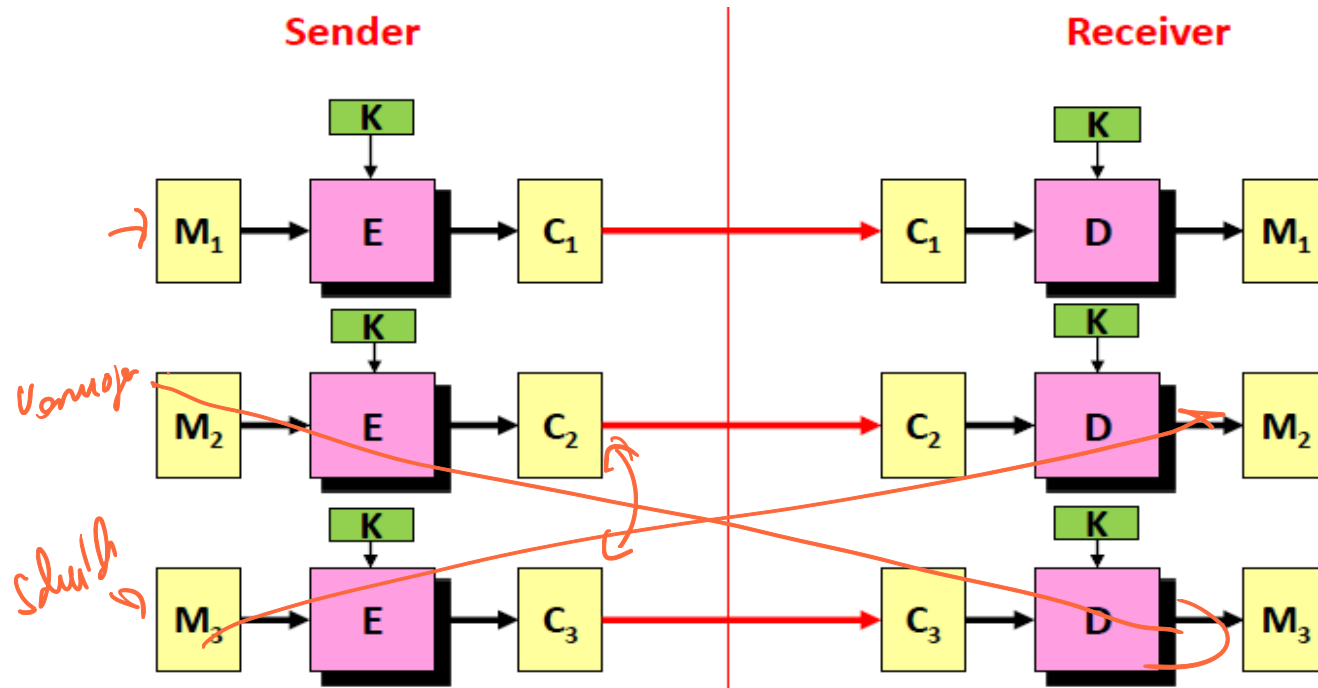


Triple (DES) für einen Block (ISO 8732)



- In Math. Kurzform: $C = E(D(E(M, K_1), K_2), K_3)$.
- **Aufgabe 8.1** Schreiben Sie die Kurzform der Entschlüsselung auf.
- Neben der oben aufgeführten Variante, gibt es noch eine mit 2 Keys: $K_1 = K_3$
- Die EDE-Form wurde wegen der Rückwärtskompatibilität zum Single-DES gewählt ($K_1 = K_2 = K_3$).
- Triple-AES ist z.Z. nicht nötig, da AES mit 256 Bits existiert.
- Der two-Key TDES (= 3DES) ist seit 2009 nicht mehr erlaubt. Sicherheit viel, viel kleiner als 112 Bit, nämlich nur ca. 57 – 60 Bit!
- Der three-Key TDES (= 3DES) ist gemäss NIST bis 2030 erlaubt, resp. sicher genug. Die Sicherheit ist ca. 112 Bit (und nicht etwa 168 Bit). Grund: Meet-in-the-middle Attacke (cf. Präsenz 5 „Kryptoanalyse“).

ECB-Modus für mehrere Blöcke

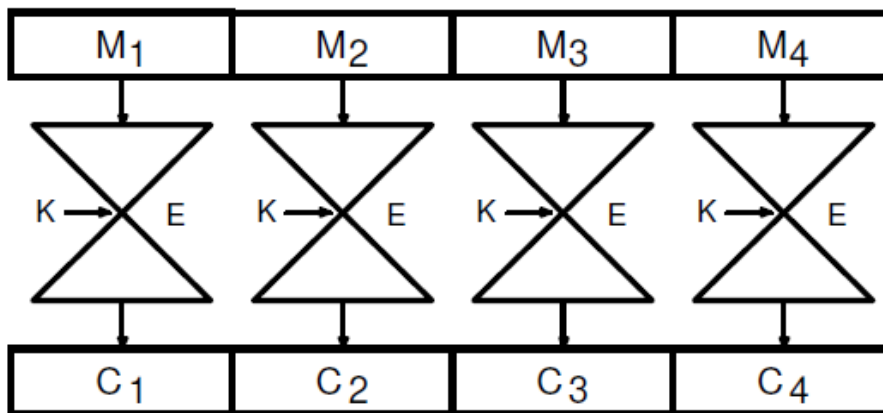
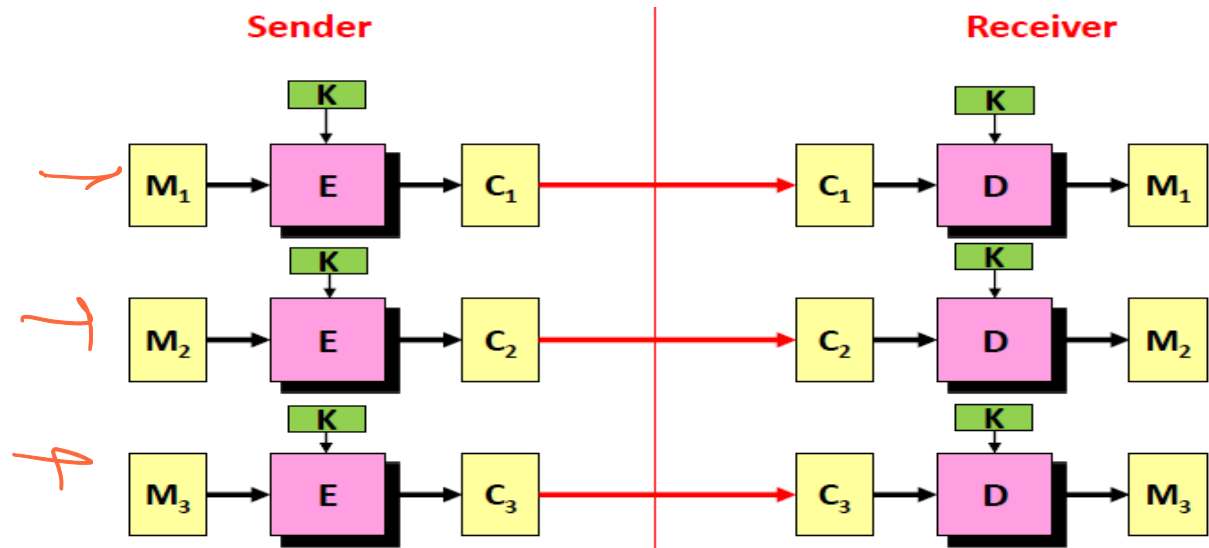


- ECB = Electronic Codebook
- In Math. Kurzform: $C_i = E(M_i, K)$
- **Aufgabe 8.2** Schreiben Sie die Kurzform der Entschlüsselung auf.
- + Die Verschlüsselung und Entschlüsselung ist parallelisierbar.
- + Eine Teilverschlüsselung und -entschlüsselung eines grossen Files oder Harddisk ist möglich.
- + Ein Verändern im Chiffretext wirkt sich beim Entschl. nur auf den direkt betroffenen Block aus.
- Gleiche Klartextblöcke werden in gleiche Chiffretextblöcke verschlüsselt.
- Ein Vertauschen von Chiffretextblöcken wird beim Entschlüsseln nicht bemerkt.
- Problem, wenn eine Meldungslänge nicht ein Vielfaches der Blocklänge ist. Lösung z.B. Padding oder mit Hilfe eines „Tricks“ (siehe CBC-Modus).

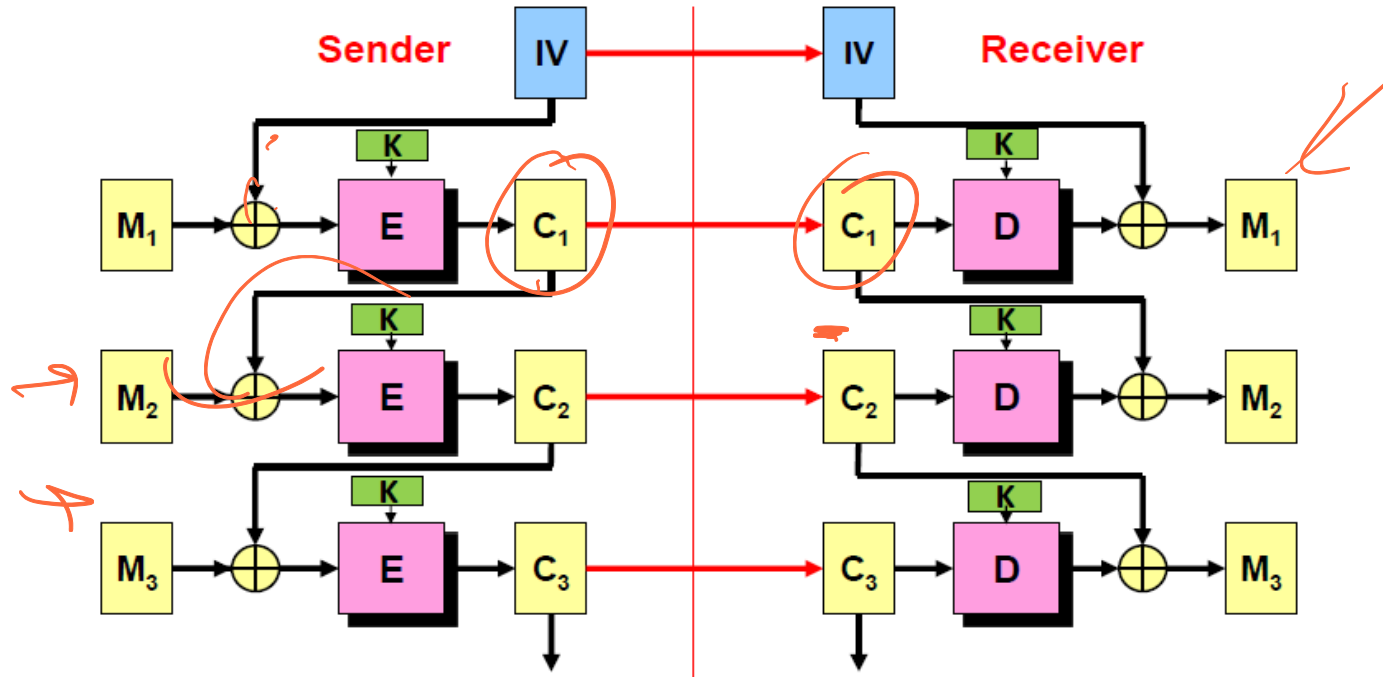
Einschub: Verschiedene Darstellungen

Sie merken sehr schnell, dass ich sowohl in den Folien und im Skript für gleiche Algorithmen verschiedene Darstellungen angebe. Das ist auch in den Büchern so: es gibt unzählige mögliche Darstellungen.

Bitte: Gewöhnen Sie sich an meine versch. Darstellungen, Sie sollten problemlos wechseln können.



CBC-Modus, CBC = Cipher Block Chaining



- In Math. Kurzform: $C_i = E(M_i \oplus C_{i-1}, K)$, mit $C_0 = \text{Initial Value}$
- **Aufgabe 8.3** Schreiben Sie die Kurzform der Entschlüsselung auf.
 - Die Verschlüsselung ist nicht parallelisierbar. **Bemerkung:** Die Entschlüsselung hingegen schon!
 - Die Teilverschlüsselung und -entschlüsselung eines grossen Files oder Harddisk ist nicht möglich.
 - Problem, wenn eine Meldungslänge nicht ein Vielfaches der Blocklänge ist.
 - + Ein Verändern im Chiffretext wirkt sich beim Entschl. auf den direkt betroffenen Block und in einem Bit des nächsten Blocks aus.
 - + Gleiche Klartextblöcke werden nicht in gleiche Chiffreblöcke verschlüsselt.
 - + Ein Vertauschen von Chiffreblöcken ist nicht mehr so einfach möglich.



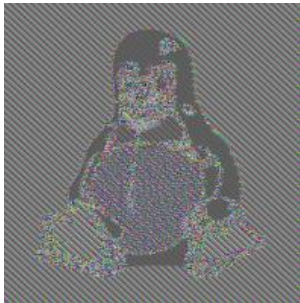
CBC-Modus, Fort.

Frage 1: Ist der Initial Value IV ein Key und für was braucht es ihn?

Antwort 1:

- Der IV = Initial Value ist kein Schlüssel und muss deshalb auch nicht geheim gehalten werden. Er kann z.B. im unverschlüsseltem Message Header mitgeschickt werden.
- Der IV dient zwei Zwecken:
 - (1) Mit dem IV ist die Verschlüsselung für jeden Block softwaretechnisch gesehen gleich. Ohne IV müsste die SW checken, ob ein Block der zu verschlüsseln ist, der erste – also ohne IV – ist oder nicht.
 - (2) Der wechselnde IV garantiert, dass die Chiffre einer Meldung, die mit dem gleichen Key wiederholt verschlüsselt wird (**Achtung:** so oder so nicht zu empfehlen) unterschiedlich werden.

Aufgabe 8.4

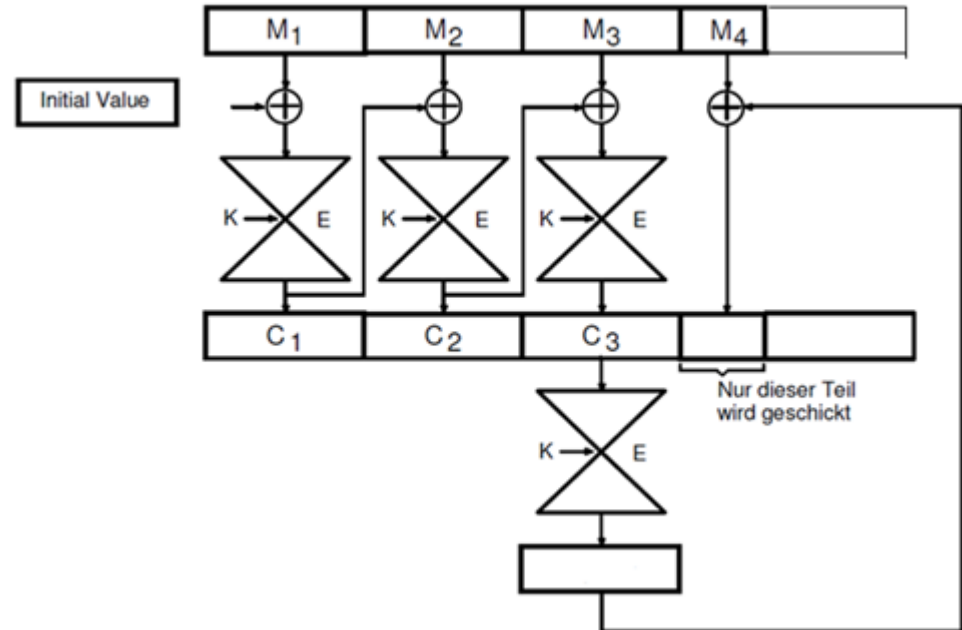
Meldung	Output Typ 1	Output Typ 2
		
Mit Modus ... verschlüsselt	CBC	ECB

CBC-Modus, Fort.

Frage 2: Was macht man, wenn die Meldungslänge kein Vielfaches der Blocklänge ist? Resp. Wie lautet nun der angesprochene „Tricks“?

Antwort 2:

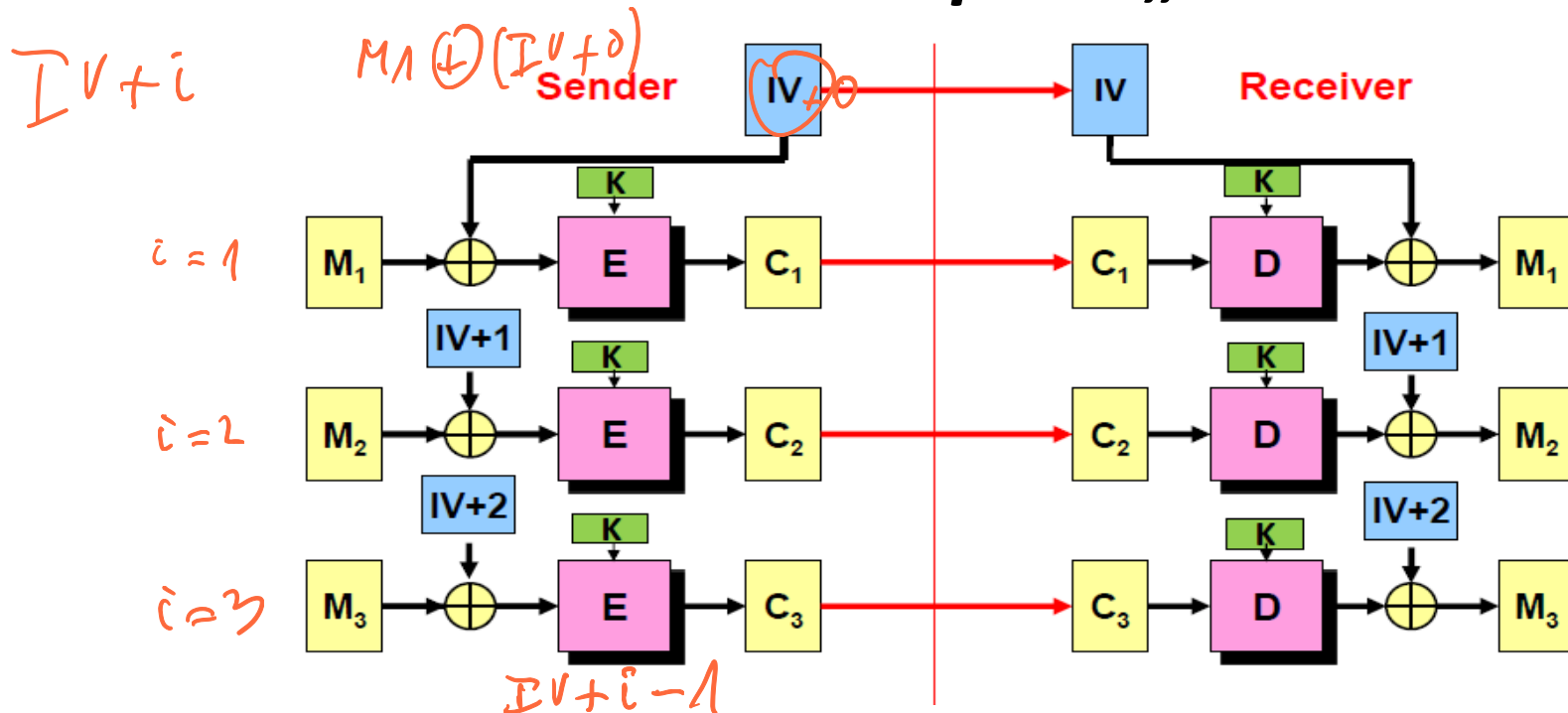
Den letzten Block speziell verschlüsseln, siehe Zeichnung.



Aufgabe 8.5 Füllen Sie die offenen Stellen des Satzes aus.

Der 4-te Block ist eine _____ mit dem nochmals verschlüsselten _____ als Schlüssel. Die Formel für den letzten Block lautet: _____. Die überflüssigen Bits werden _____. Es ist zu beachten, dass bei der Entschlüsselung die Formel für den 4-ten Block _____. D.h. auch bei der Entschlüsselung muss für die Schlüsselerzeugung eine _____ gemacht werden!

Didakt. Gründe: ein nicht publ. „Mix Modus“



Dieser nicht publizierte „Mix-Modus“ vereinigt die Vorteile von ECB und CBC.

- + Die Verschlüsselung und Entschlüsselung ist parallelisierbar.
- + Eine Teilverschlüsselung und -entschlüsselung eines grossen Files oder Harddisk ist möglich.
- + Gleiche Klartextblöcke werden **nicht** in gleiche Chiffpratblöcke verschlüsselt.
- + Ein Vertauschen von Chiffpratblöcken ist nicht mehr so einfach möglich.

bei 12430

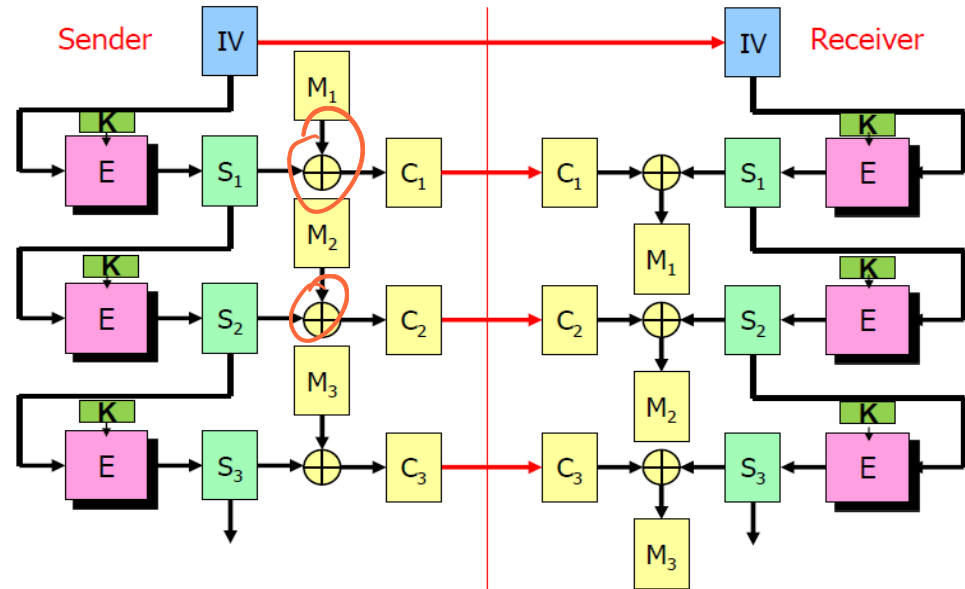
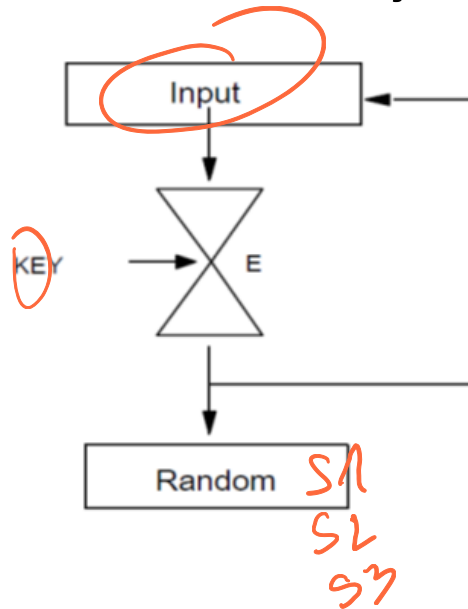
Aufgabe 8.6 Schreiben Sie die Kurzform der Ver- und Entschlüsselung auf.

Weiterentwicklung: Der CTR Mode (CTR = Counter Mode), cf. Kap. 8.3.

Kap. 8.3

VERSCHLÜSSELUNG MIT BLOCKCHIFFREN IM STREAMCIPHERMODUS

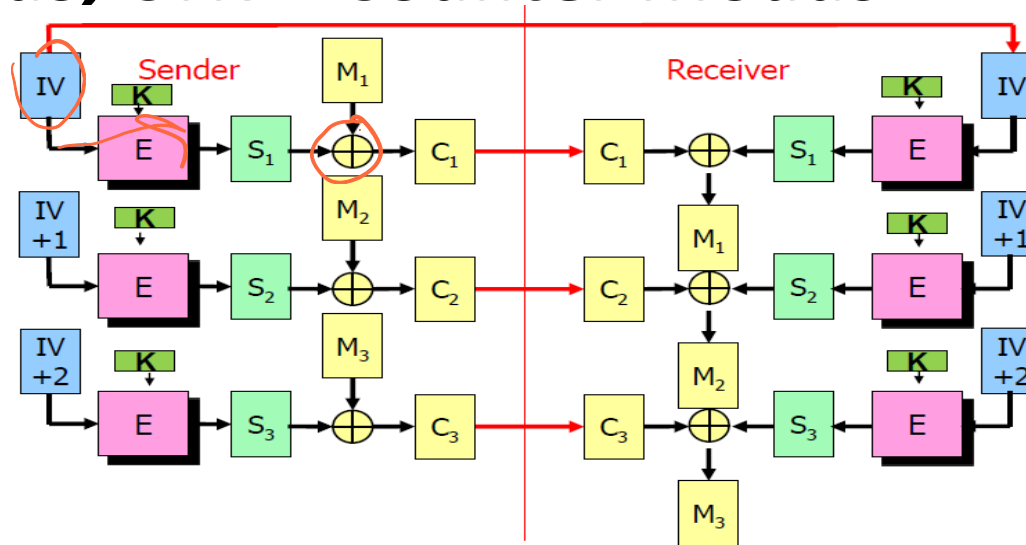
OFB-Modus, OFB = Output Feedback



- Die Idee ist nun, dass die Blockchiffre nicht für die Verschlüsselung, sondern „nur“ als Pseudo Zufallszahlengenerator, also für die Schlüsselerzeugung gebraucht wird.
- Eine Stromchiffre hat immer den Vorteil, dass die Bedingung „ein Vielfaches der Blocklänge“ nicht erfüllt sein muss.
- In Math. Kurzform: $C_i = S_i \oplus M_i$, mit $S_i = E(S_{i-1}, K)$ und $S_0 = \text{Initial Value}$
- **Aufgabe 8.7** Schreiben Sie die Kurzform der Entschlüsselung auf. Und vervollständigen Sie den Satz: Auch beim Entschlüsseln ist die Blockchiffre im _____ Modus.
- Irgendeinmal wiederholen sich die Schlüssel, dies geschieht spätestens nach 2^n Durchgängen. Zweimal der gleiche Schlüssel ist bei einer Stromchiffre der Supergau.
- Auf Grund von Wahrscheinlichkeitstheoretischen Gegebenheiten besteht die mittlere Zykluslänge aus $\sqrt{2^n}$, wenn n die Blockgröße ist. D.h. bei $n = 64$ ist das $\sqrt{2^{64}} = 2^{32}$, also ca. 4 Milliarden, also für heutige Verhältnisse relativ klein. Bei $n = 128$ ist das $\sqrt{2^{128}} = 2^{64}$.

CTR-Modus, CTR = Counter Modus

Stromchiffre



- In Anlehnung an den nicht publizierten, aber aus didaktischen Gründen behandelten „Mix Modus“ (eine Blockchiffre) erfüllt der CTR – Modus als Stromchiffre die gleichen guten Eigenschaften:
 - + Die Verschlüsselung und Entschlüsselung ist parallelisierbar.
 - + Eine Teilverschlüsselung und -entschlüsselung eines grossen Files oder Harddisk ist möglich.
 - + Vorausberechnung vom Schlüsselmaterial ist möglich.
 - + Gleiche Klartextblöcke werden nicht in gleiche Chiffpratblöcke verschlüsselt.
 - + Ein Vertauschen von Chiffpratblöcken ist nicht mehr so einfach möglich.
- In Math. Kurzform: $C_i = S_i \oplus M_i$, mit $S_i = E(IV + (i - 1), K)$
- Auch hier gilt, dass bei der Entschlüsselung die Blockchiffre im Encrypt Modus ist.

Wie schon zu Beginn des Kap. 8 erwähnt, gibt es noch viele weitere Modi, insbesondere spannend sind die Kombi Modi, die Verschlüsseln und MAC'en in einem Aufruf machen können.

Fehlerfortpflanzung bei der Entschlüsselung

Die grosse Preisfrage:

Was passiert, wenn im Chiffretext **ein einziges Bit** gekippt wird?

Die spontane Antwort, die i.d.R. gegeben wird:

Alles ist nun verändert, da kann nichts mehr entziffert werden!

Nur das ist grundlegend falsch:

Grün = unveränderte Bits

Dicke Umrandung = ein Block mit Blocklänge des Blockchiffriers.

Fehler im Chiffretext																	
Klartext ECB-Modus																	
Klartext CBC-Modus																	
Klartext „Mix“-Modus (*)																	
Klartext OFB-Modus (**)																	
Klartext CFB-Modus (***)																	
Klartext CTR-Modus (**)																	

(*) Kein offizieller Modus, ist aber eine Art ECB Modus.

(**) Stromchiffren

(***) Im JS Skript, aber nicht in den Folien behandelt, nicht Prüfungsstoff.

Bemerkung: Im JS Skript, Kap. 8.1.6 & 23 hat es Beispiele dazu.

Kap. 8.2

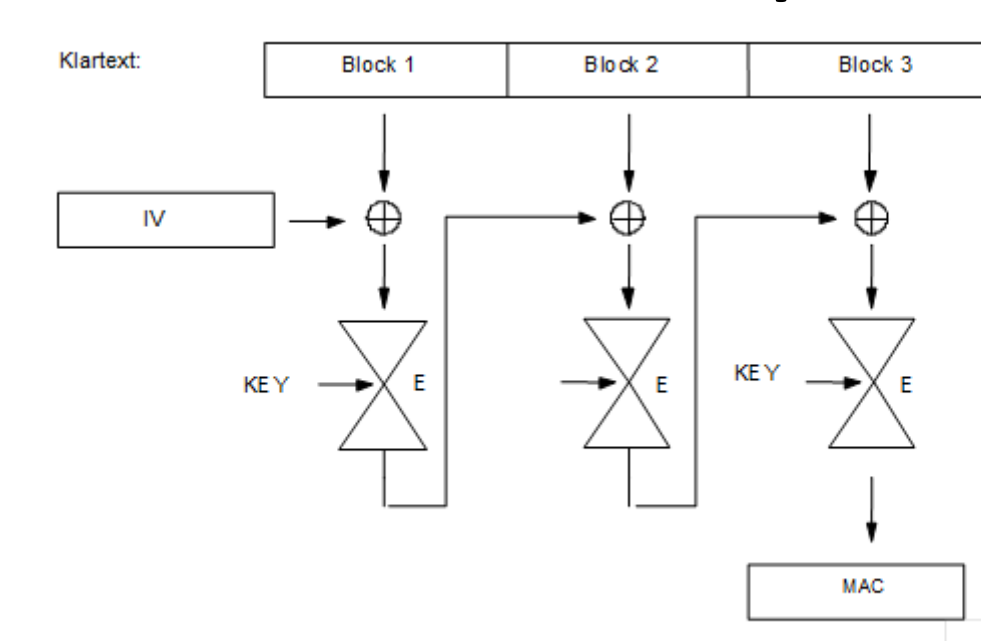
INTEGRITÄTSSCHUTZMECHANISMEN

MIT ...

- ... BLOCKCHIFFREN
- ... HASHFUNKTIONEN

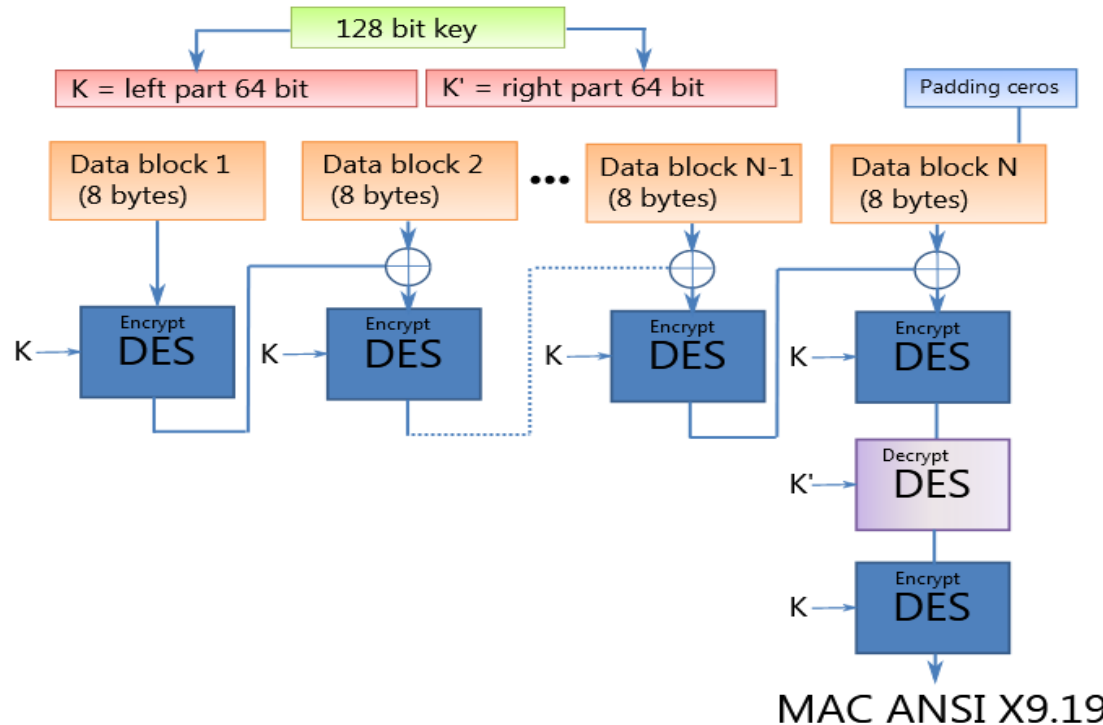
Dieses Kapitel wird ev. aus zeitlichen und inhaltlichen Gründen als Teil der Präsenz 3 besprochen.

Der CBC-MAC ANSI X9.9, resp. ISO 8730



- Der CBC-MAC, wie oben dargestellt, ist die Mutter aller Integritätsschutzmechanismen mit Blockchiffren. Er wird in allen Lehrbüchern dargestellt.
- Er hat gewisse (theoretische) Schwächen und ist eigentlich als Standard schon länger zurückgezogen.
- Es gibt diverse Anpassungen (cf. nächste Folie), gemäss ISO 9797-1 → ist aber auch als Standard zurückgezogen.
- In der Industrie gibt es viele individuelle weitere leicht angepasste Versionen.
- All diese Formen werden aber nach wie vor konkret verwendet. Dies gemäss Aussagen von Securityspezialisten, die „nahe am Geschehen sind“, so im Sinne „never change a winning team“.

Der CBC-MAC nach ISO 9797-1, resp. ISO 9807

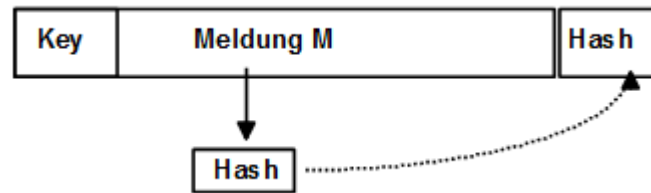


- In ISO 9797-1 wurden 6 Varianten definiert, die oben dargestellte ist eine dieser Varianten und wurde als ANSI X9.19 Standard aufgeführt.
- Es gibt auch neuere Varianten, eben vor allem auch im Zusammenhang von kombinierten Modi.

Andere Möglichkeiten des Integritätsschutzes

Schon früh kam der Wunsch auf, (grosse) Files gegen Abändern zu schützen. Oft enthielten diese Files keine Daten, die gegen Abhören geschützt werden mussten. D.h. eine Verschlüsselung – und damit z.B. Blockchiffren – waren nicht nötig. Dazu kommt, dass dieser Integritätsschutz nicht rechenintensiv sein sollte.

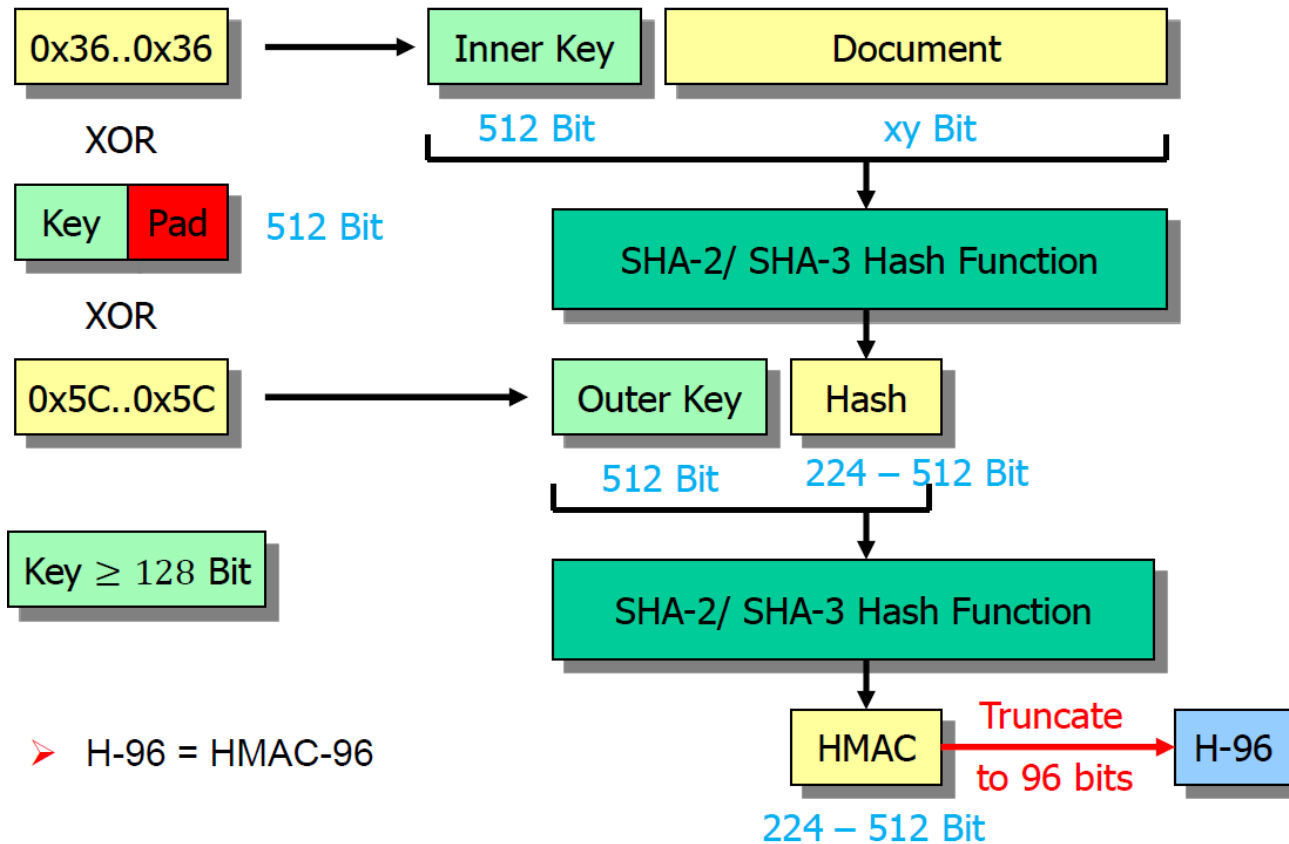
Da kam die Idee schlüssellose Hashfunktionen zu nehmen und vor die Meldung M einen Schlüssel vorzuhängen und Ganze zu hashen.



Man fand schnell Angriffe, und versuchte es damit, vorne und hinten einen Schlüssel anzuhängen. Auch so fand man Angriffe, und man definierte HMAC nach RFC 2104.

Mehr dazu und zur Geschichte vom HMAC (cf. nächste Folie) in der Präsenz 3.

HMAC (HashMAC) nach RFC 2104



Bemerkungen:

- Grundsätzlich kann der Key bis 512 Bit gross sein.
- Warum mit den Konstanten 36 und 5C gepadded wird, ist im RFC nicht ersichtlich.
- Gemäss BSI sind als Hashfunktionen nur noch SHA2 & 3 erlaubt.
- Wird oft in den Internetprotokollen wie TLS, IPSEC usw. verwendet, wobei in den neueren Versionen nun auch Kombi Modi wie GCM verwendet werden, cf. JS Skript, Kap. 8.6.
- Das Abschneiden auf 96 Bit ist eigentlich eine Schwächung, cf. JS Skript, Kap. 8.2.4.

Reihenfolge von Verschlüsseln und MAC'en

Bez. der Reihenfolge von Verschlüsseln und MAC'en, resp. Signieren ist schon länger eine relativ hitzige Debatte am Laufen, cf. JS Skript Kap. 8.2.5.

Wenn auf dem Application Layer die Kryptographie eingesetzt wird, lautet die klassische Variante ist:

Über den Klartext den MAC oder die Signatur berechnen und dann Verschlüsseln.

Das sieht dann in einer Übersichtszeichnung wie folgt aus:

Header	Body oder Meldungsinhalt	Trailer
Mit MAC geschützt		MAC ist im Trailer
unverschlüsselt	verschlüsselt	unverschlüsselt
Enthält z.B. Routingdaten, Datum, Zeit & IV	Enthält die gesamten Nutzdaten	Enthält z.B. MAC und/oder verschlüsselte Schlüssel.
Unverschlüsselt und mit MAC geschützt.	Verschlüsselt und mit MAC geschützt.	In der Regel weder noch. (**)

(**) Es gibt aber Protokolle, z.B. der Encapsulating Security Payload (ESP) bei IPSec, wo ein Teil des Trailers verschlüsselt und mit einem MAC versehen ist. In SSL 2.0 wird der MAC ebenfalls verschlüsselt der Grund liegt hier darin, dass in SSL 2.0 eine schwache MAC-Konstruktion gewählt wurde.

Kap. 8.4

RANDOMFUNKTIONEN

MIT ...

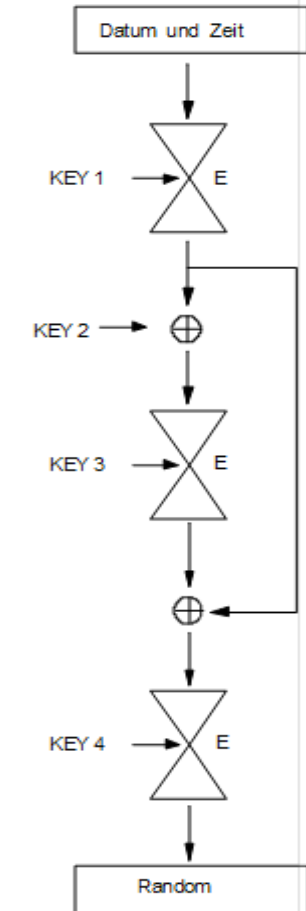
- ... BLOCKCHIFFREN
- ... HASHFUNKTIONEN

Dieses Kapitel wird ev. auch (wird sich im Verlaufe der Präsenz von Präsenz 2 automatisch ergeben) aus zeitlichen und inhaltlichen Gründen als Teil der Präsenz 3 besprochen.

Beispiel einer Randomfkt. mit Blockchiffren

Bemerkungen:

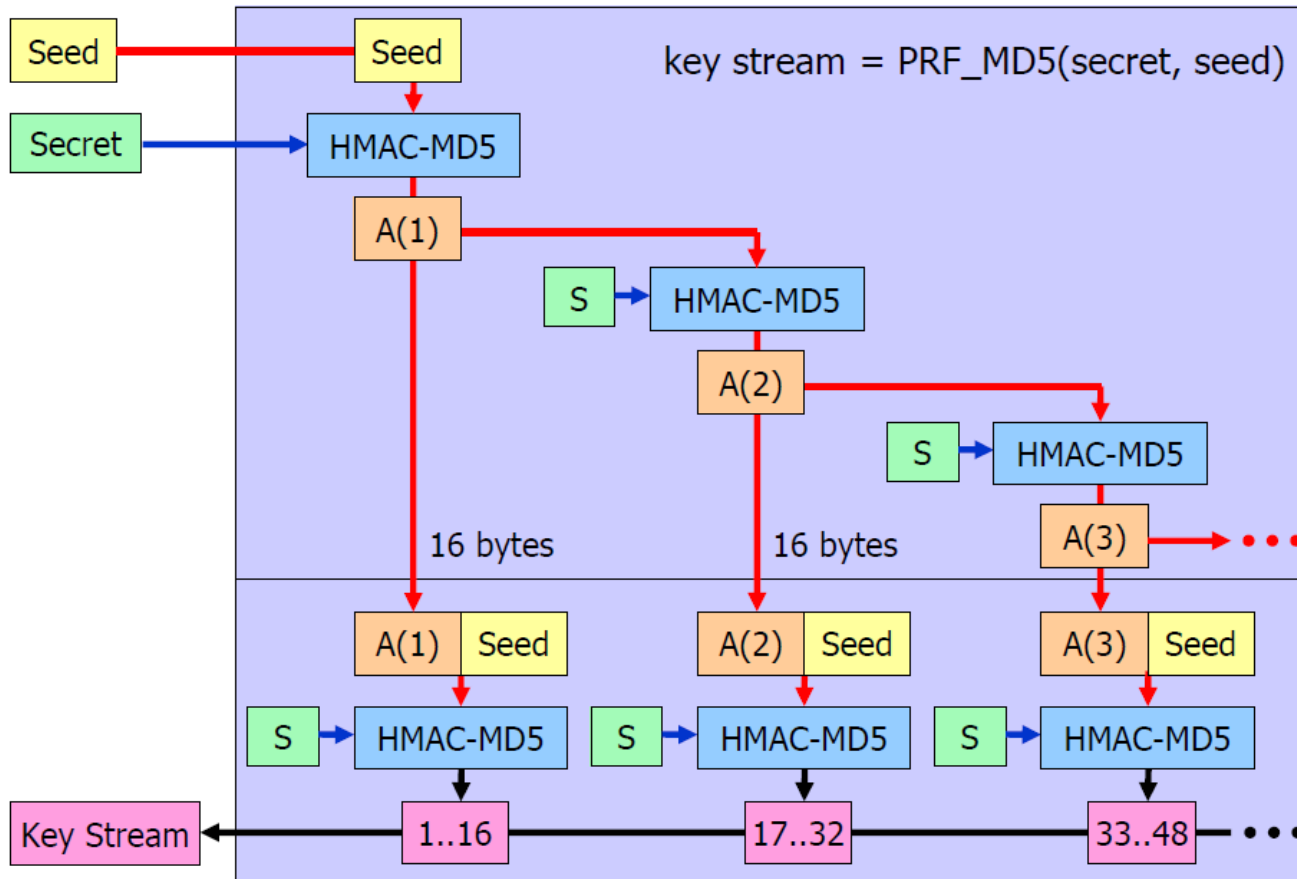
- Die Version mit nur 2 verschiedenen Schlüsseln (Key 3 = Key 4 = Key 1) war früher im Annex von ISO 8732 standardisiert. Der Standard – und damit der Annex – wurde zurückgezogen.
- **Nie** Hardware Zufallszahlengeneratoren (z.B. Rauschdioden) alleine verwenden.
- Wenn ein Hardware Zufallszahlengenerator (z.B. Rauschdiode) zur Verfügung steht, dann soll er nur benutzt werden, um den obigen Input zu verbessern.
- **Vorschlag:** Input = (Datum || Zeit) \oplus Output der „Rauschdiode“
- **Analyse des Vorschlags:** Sollte die Rauschdiode kleine Zyklen liefern, ist der Input immer noch so gut, wie (Datum || Zeit) alleine.
- Wird eine solche Funktion zur Erzeugung von Schlüsseln gebraucht, dann nennt man sie auch **KDF (Key Derivation Function)**.
- Im Rahmen des Key Managements (Präsenz 4) werden wir auf diese Funktion zurückkommen.



Beispiel einer Randomfkt. mit Hashfkt. (*)

(*) Nicht Prüfungsstoff, nur Wissen, dass es solche Randomfunktionen gibt.

Pseudo Random Function (PRF)



Basis-Test Präsenz 2 zu Kap. 8

Aussage	Richtig oder falsch?	Begründung
Blockchiffren haben vielfältige Einsatzmöglichkeiten.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Die zwei Betriebsmodi ECB und CBC von Blockchiffrieren haben die gleichen Eigenschaften.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Der CTR Modus verbindet die guten Eigenschaften von ECB und CBC.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Im OFB und CTR Modus wird nur der Schlüssel mit einer Blockchiffre erzeugt. Die Verschlüsselung ist eine Stromchiffre.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Wird bei einer Meldung, die mit dem CBC Modus verschlüsselt ist, ein Bit im Chifftrat gekippt, dann ist ab diesem Block die entschlüsselte Meldung völlig kaputt.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Der CBC-MAC ist ein Modus, um die Meldungsintegrität zu gewähren.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Der HMAC ist ein weiterer Modus, wie man mit Blockchiffren Integrität gewähren kann.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Eine Triple-DES Entschlüsselung mit 3 verschiedenen Schlüsseln kann mit $M = D(D(D(C, K_1), K_2), K_3)$ beschrieben werden. Dabei ist M der Klartext und C das Chifftrat.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	

bis 17h45

ZUSAMMENFASSUNG DER VORBEREITUNG (PRÄSENZ 0) UND VON DEN PRÄSENZEN 1 & 2

Zusammenfassung I

- Vertraulichkeit/Geheimhaltung
 - Massnahmen
 - symmetrische Verschlüsselung
 - Verfahren: Strom- und Blochchiffren
 - Algorithmen: DES(*); 3-DES(**), AES, IDEA usw.
 - Modi: ECB(*), CBC, OFB, CFB, CTR usw.
 - (*) „darf“ nicht mehr verwendet werden.
 - (**) „sollte“ nicht mehr verwendet werden.
 - asymmetrische Verschlüsselung
 - Verfahren: diskrete Logarithmen, diskretes Wurzelziehen
 - Algorithmen: ECC (Elliptic Curve), RSA usw.
 - Modi: nichts Vergleichbares wie für symmetrische Verfahren.
 - Verschlüsselung schützt nur vor Abhören!!
- Daten-Integrität/Daten-Authentizität
 - Massnahmen:
 - MAC-Berechnung (symmetrisch)
 - Schützt vor Verfälschen und Einfügen
 - CBC-MAC (auf Blochchiffren beruhend, cf. Kap. 8.2)
 - Vor allem in „Zahlungswelt“ verwendet
 - HMAC (auf Hashfunktionen beruhend)
 - Vor allem in der „Netzwerkwelt“ verbreitet.

Zusammenfassung II

- Daten-Integrität/Daten-Authentizität, Fortsetzung
 - Massnahmen, Fortsetzung:
 - Digitale Signatur (asymmetrisch)
 - Schützt vor Verfälschen, Einfügen und Abstreiten eine Meldung geschickt zu haben.
 - Verfahren: diskrete Logarithmen, diskretes Wurzelziehen
 - Algorithmen: ECC (Elliptic Curve), RSA, DSA, Schnorr usw.
- Replay und Delete
 - Massnahmen:
 - Führen einer Sequenznummer und MAC-Berechnung (symmetrisch)
 - Schützt zusätzlich vor Replay und Delete
 - Führen einer Sequenznummer und digitale Signatur (asymmetrisch) → wird in dieser Form aber eher nicht gemacht.
 - Schützt ebenfalls zusätzlich vor Replay und Delete
- Non rep. of Receipt und Masquerade (Benutzer- resp. Instanzauthentizität, Authentisierung)
 - Massnahmen:
 - Krypt. Mechanismus (MAC oder digitale Signatur) in ein Protokoll eingebettet.

**EINFÜHRUNG INS
WISSENSCHAFTLICHE ARBEITEN IN
DER KRYPTOLOGIE (*)
NICHT PRÜFUNGSSTOFF**

Einleitung

Das wissenschaftliche Arbeiten – z.B. Schreiben einer Semesterarbeit (SecProj oder WiPro), der BSc Abschlussarbeit (BAA), später die Masterarbeit usw. – ist schlussendlich das Ziel eines Hochschulstudiums. Das wissenschaftliche Arbeiten unterscheidet sich gänzlich vom Lernen eines Modulinhalts.

- Die Aufgabenstellung ist meistens sehr offen gestellt. → Thesis, Annahme, (vage) Vermutung.
- Die Aufgabenstellung kann sich aufgrund erster Erkenntnisse ändern.
- Das Ergebnis kann grundsätzlich offen sein (z.B. es kann auch sein, dass man mit der Vermutung falsch liegt und schlussendlich zeigt, dass die Vermutung nicht stimmt).
- Es gibt nicht mehr „löse ein paar Aufgaben, in der Prüfung kommt dann eine ähnliche, dann bestehst du die Prüfung.“
- **Und insbesondere rechnet Ihnen kein Dozent die Lösung vor, es gibt auch keine vorgefertigte Musterlösung.**

Nur das wissenschaftliche Arbeiten – und erst recht dasjenige in Kryptologie – muss auch gelernt und geübt werden. Das möchte ich an dieser Stelle von mehr oder weniger offenen Aufgaben in Kryptologie tun.

Sie sollen also ein von mir gestelltes Problem selbstständig analysieren, eine Lösungsstrategie entwickeln, ggf. weitere Informationen recherchieren und schlussendlich ein Resultat (inkl. Herleitung und Beschreibung) präsentieren.

Die Aufgabe ist **freiwillig**, man kann mich immer für Fragen, Zwischenkontrollen oder einer Schlussbesprechung kontaktieren.

Josef Schuler, Dozent für Mathematik und Kryptologie, josef.schuler@hslu.ch

Artikel zu «No Key Verschlüsselung»

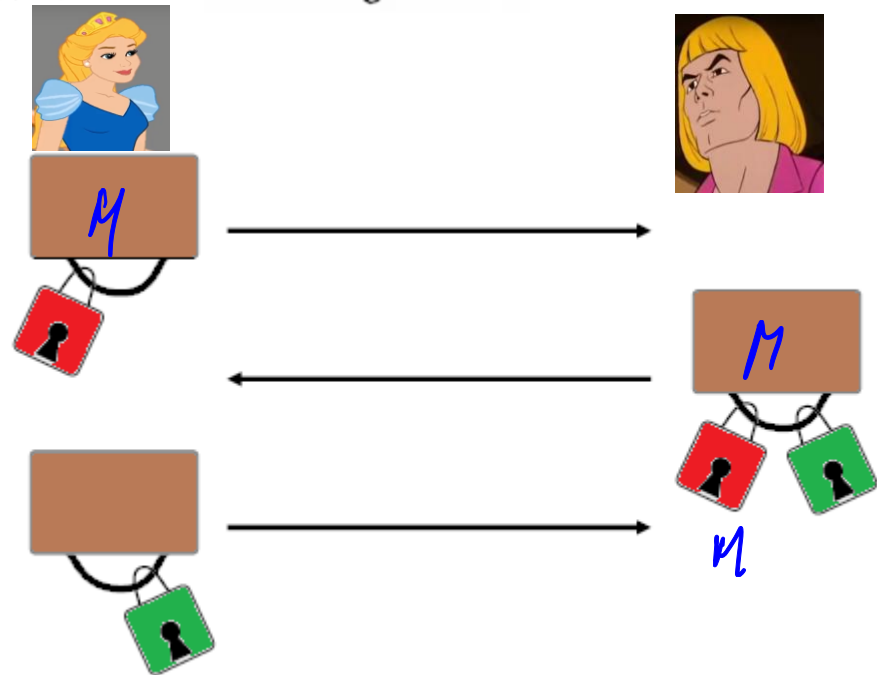
Eine Nachricht geheim über eine öffentliche Leitung verschicken, ohne dass Sender und Empfänger einen Schlüssel vereinbart haben:

Sender und Empfänger schicken die Nachricht dreimal hin und her. Auf ihrem ersten Weg vom Sender zum Empfänger ist die Nachricht mit dem geheimen Schlüssel des Senders verschlüsselt. Auf dem Rückweg ist sie doppelt verschlüsselt – zusätzlich mit dem geheimen Schlüssel des Empfängers. Auf dem erneuten Weg zum Empfänger ist sie nur noch einfach verschlüsselt – der Sender hat seine Verschlüsselung rückgängig gemacht. Wenn der Empfänger die Nachricht nun erhält, macht er seinerseits seine Verschlüsselung rückgängig und hat den Klartext vorliegen.

Auf ihrem Weg hin und her ist die Nachricht stets verschlüsselt, sodass kein Dritter sie lesen kann. Die Bezeichnung »No Key« ist insofern nicht ganz richtig, aber Sender und Empfänger verwenden nur ihre jeweils eigenen Schlüssel und vereinbaren keinen gemeinsamen Schlüssel.

Umsetzung mit Koffer, Vorhängeschlösser VHS und Kurier:

- Alice hängt rotes VHS an.
- Bob hängt grünes VHS an.
- Alice nimmt rotes VHS weg.
- Bob nimmt grünes VHS weg und kann den Koffer öffnen.



Detaillierte Aufgabenstellung

- (1) Zum allgemeinen Verschlüsselungsprotokoll:
 - i. Zeichnen Sie das allgemeine Verschlüsselungsprotokoll auf.
 - ii. Formulieren Sie die allgemeinen Bedingungen, so dass das Protokoll funktioniert.
- (2) Nun verwenden Sie a) eine Blockchiffre resp. b) eine Stromchiffre und kommentieren Sie das Protokoll, bezüglich Machbarkeit. Wenn es machbar ist, dann analysieren Sie es, ob es auch sicher ist.
 - a) Bei Verwendung einer Blockchiffren
 - b) Bei Verwendung der Stromchiffren

Tipp: Verwenden Sie die folgenden Bezeichnungen für das allgemeine Verschlüsselungsprotokoll, resp. für die Blockchiffre a).

- Verschlüsselung der Meldung M mit Schlüssel K_i ergibt Chiffre $C_i = E(M; K_i)$. Dabei ist M entweder die ursprüngliche Meldung M oder dann ein Chiffre C_i , d.h. $C_j = E(C_i; K_j)$.
- Somit ist dann eine Doppelverschlüsselung z.B. $C_2 = E(C_1; K_2) = E(E(M; K_1); K_2)$
- Entschlüsselung des Chiffres C_i mit Schlüssel K_i ergibt Meldung M : $M = D(C_i; K_i)$. Dabei ist M entweder die ursprüngliche Meldung M oder dann ein Chiffre C_i .
- Das Verschlüsseln und dann das Entschlüsseln muss kombiniert werden.
- Bei der Stromchiffre b) kann direkt mit \oplus gearbeitet werden.
- Neben den obigen Bezeichnungen, benutzen Sie – insbesondere für (1) i. – die Vorlage auf der nächsten Folie.

Vorlage Protokollablauf

Alice mit K_1	unsichere Leitung	Bob K_2
	----->	
	<-----	
	----->	

LÖSUNGEN DER AUFGABEN

Aufgabe 7.0

Ja, klar gibt es einen Haken. Das Problem ist, den Schlüssel k dem Empfänger „unendlich“ sicher zu übermitteln. Wenn ich das ja könnte, dann könnte ich ihm ja auch den zu schickenden Buchstaben (ohne zu verschlüsseln) „unendlich“ sicher zusenden.

Aufgabe 7.1 CE

Aufgabe 7.2 Es gibt immer Null.

Aufgabe 7.3 Nur im Skript enthalten.

Aufgabe 7.4 Nur im Skript enthalten.

Aufgabe 7.5 Im Skript ist eine leicht andere Fragestellung, und sie enthält andere Werte.

- a) Key3 ist das XOR von Key1 und Key2, also: $Key3 = Key1 \oplus Key2 = 8E \oplus 58 = D6$
- b) Direkt gerechnet: Chiffre = $32 = Text \oplus Key3 \Rightarrow Text = 32 \oplus Key3 = 32 \oplus D6 = E4$
Oder: $32 = Text \oplus \underbrace{(Key1 \oplus Key2)}_{Key3} \Rightarrow Text = 32 \oplus Key3 = 32 \oplus D6 = E4$

Aufgabe 7.6

- a) Schutz gegen das Abhören der Meldung.
- b) Stromchiffre ist hauptsächlich ein Schutz gegen intelligente Gegner, wie z.B. Hacker, Geheimdienste usw., die einen passiven Angriff durchführen. **Grund:** Abhören ist ein passiver Angriff und in diesem Sinne sind die Angreifer passive Angreifer.
- c) Symmetrische Verschlüsselung.

Aufgabe 7.7 a) Offensichtlich ging sowohl beim Sender wie Empfänger etwas schief.

b) **Empfänger:**

So wie es aussieht ist beim Empfänger $M_1 \oplus M_2$ als Resultat herausgekommen. Das Resultat ist – möglicherweise – aus der Berechnung von $C_1 \oplus C_2$ entstanden.

Sender:

Beim Sender wurde (sehr wahrscheinlich) mit dem gleichen Schlüssel verschlüsselt, also z.B.

$$C_1 = M_1 \oplus K_1 \quad \text{und} \quad C_2 = M_2 \oplus K_1$$

Oder

$$C_1 = M_1 \oplus K_2 \quad \text{und} \quad C_2 = M_2 \oplus K_2$$

Mit diesen Ausgangslagen kann man nun die Puzzleteile zusammenstellen:

$$C_1 \oplus C_2 = (M_1 \oplus K_1) \oplus (M_2 \oplus K_1) = (M_1 \oplus M_2) \oplus (K_1 \oplus K_1) = M_1 \oplus M_2$$

Oder

$$C_1 \oplus C_2 = (M_1 \oplus K_2) \oplus (M_2 \oplus K_2) = (M_1 \oplus M_2) \oplus (K_2 \oplus K_2) = M_1 \oplus M_2$$

OK, das sind Vermutungen, nur eine genaue Analyse des Programmiercodes kann die def. Antwort liefern. Trotzdem können wir uns fragen: Wie konnte das passieren?

Sender: Z.B. Programmierfehler; es wurde in einem Falle auf einen falschen Schlüsselplatz zugegriffen.

Oder

Man hatte in beide Schlüsselplätze den gleichen Schlüssel K_1 oder K_2 anstatt K_1 und K_2 geladen, d.h. man hatte in der Schlüsseldatei $K_1||K_1$ oder $K_2||K_2$ anstatt $K_1||K_2$.

Empfänger: Ein Programmierfehler ist hier die wahrscheinlichste Ursache.

Schlussbemerkung:

Es ist natürlich relativ unwahrscheinlich, dass bei Sender und Empfänger gleichzeitig so schwere Fehler passieren, aber unmöglich ist es nicht.

Aufgabe 8.1 $M = D(E(D(C, K_3), K_2), K_1).$

Aufgabe 8.2 $M_i = D(C_i, K)$

Aufgabe 8.3 $M_i = D(C_i, K) \oplus C_{i-1}$, mit $C_0 = \text{Initial Value} = IV$

Aufgabe 8.4

Meldung	Output Typ 1	Output Typ 2
Mit Modus ... verschlüsselt	CBC (siehe später CTR, OFB, CFB, „Mix“-Modus)	ECB

Aufgabe 8.5

Der 4-te Block ist eine Stromchiffre mit dem nochmals verschlüsselten Chifftrat C_3 als Schlüssel. Die Formel für den letzten Block lautet: $C_4 = E(C_3, K) \oplus M_4$. Die überflüssigen Bits werden einfach abgeschnitten.

Es ist zu beachten, dass bei der Entschlüsselung die Formel für den 4-ten Block nicht ändert. D.h. auch bei der Entschlüsselung muss für die Schlüsselerzeugung eine Verschlüsselung gemacht werden!

Aufgabe 8.6 $C_i = E(M_i \oplus (IV + i - 1), K); \quad M_i = D(C_i, K) \oplus (IV + i - 1)$

Aufgabe 8.7 $M_i = S_i \oplus C_i$, mit $S_i = E(S_{i-1}, K)$ und $S_0 = \text{Initial Value}$

Auch beim Entschlüsseln ist die Blockchiffre im Encrypt Modus.

Basis-Test Präsenz 2 zu Kap. 7.1

Aussage	Richtig oder falsch?	Begründung
Als Blockchiffre ist AES mit 256 Bit Schlüssellänge zu empfehlen.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	Es gibt zwar noch andere Blockchiffren, aber der AES – wenn möglich mit 256 Bit – sind vorzuziehen.
Ein Tripple-DES mit 3 versch. Schlüsseln hat eine Sicherheit von 112 Bit.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	Dies wegen einer sog. Meet-in-the-middle Attacke.
Wird im Input einer Blockchiffre ein Bit geändert, dann ändert – bei gleichem Schlüssel – die erste Hälfte der Bits des Chiffrats.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Es ändern im stat. Mittel die Hälfte aller Bits.
Wird ein Schlüsselbit einer Blockchiffre ein Bit geändert, dann ändern sich – bei gleichem Input – im stat. Mittel die Hälfte der Bits des Chiffrats.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Das Verschlüsseln und Entschlüsseln dauern beim DES gleich lang.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Das Verschlüsseln und Entschlüsseln dauern beim AES gleich lang.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Das Entschlüsseln braucht beim AES etwa doppelt so lange wie das Verschlüsseln.
Beim AES-256 dauert die Verschlüsselung etwa doppelt so lange wie beim AES-128.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Es braucht nur ca. 40% mehr Zeit. Dies weil statt 10 neu 14 Runden durchgeführt werden.
Lightweight Cryptography wird in Zukunft immer wichtiger.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	Vornehmlich Internet der Dinge IoT.
Es gibt noch keine Lightweight Crypto-Algorithmen.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	PRESENT ist eine Lightweight Blockchiffre.

Basis-Test Präsenz 2 zu Kap. 7.2

Aussage	Richtig oder falsch?	Begründung
Das wirkliche „Arbeitspferd“ der Kryptologie sind die Stromchiffren.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Nein, das gilt für die Blockchiffren.
Bei Stromchiffren ist die Verschlüsselung und die Entschlüsselung exakt die gleiche Operation.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	In beiden Fällen das XOR.
Es gibt mehrere Stromchiffre Verschlüsselungsalgorithmen.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Es gibt aber unzählige Möglichkeiten den Schlüsselstrom zu erzeugen.
Stromchiffre und One-Time-Pad sind identische Begriffe.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Ein OTP ist eine Stromchiffre, aber nicht umgekehrt.
Mit Stromchiffren kann ich neben dem Verschlüsseln auch die Integrität einer Meldung gewähren.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Stromchiffren bieten keine Integrität.
$12 \oplus 36 = 48$	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	$12 \oplus 36 = 24$

Basis-Test Präsenz 2 zu Kap. 8

Aussage	Richtig oder falsch?	Begründung
Blockchiffren haben vielfältige Einsatzmöglichkeiten.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	Es gibt – grob gesagt – 4 unterschiedliche Einsatzmöglichkeiten für Blockchiffren.
Die zwei Betriebsmodi ECB und CBC von Blockchiffrieren haben die gleichen Eigenschaften.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	
Der CTR Modus verbindet die guten Eigenschaften von ECB und CBC.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Im OFB und CTR Modus wird nur der Schlüssel mit einer Blockchiffre erzeugt. Die Verschlüsselung ist eine Stromchiffre.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Wird bei einer Meldung, die mit dem CBC Modus verschlüsselt ist, ein Bit im Chifftrat gekippt, dann ist ab diesem Block die entschlüsselte Meldung völlig kaputt.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Der CBC erholt sich wieder bei der Entschlüsselung.
Der CBC-MAC ist ein Modus, um die Meldungsintegrität zu gewähren.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Der HMAC ist ein weiterer Modus, wie man mit Blockchiffren Integrität gewähren kann.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Der HMAC ist eine MAC Konstruktion mit einer Hashfunktion wie SHA-3.
Eine Triple-DES Entschlüsselung mit 3 verschiedenen Schlüsseln kann mit $M = D(D(D(C, K_1), K_2), K_3)$ beschrieben werden. Dabei ist M der Klartext und C das Chifftrat.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Korrekt ist: $M = D(E(D(C, K_3), K_2), K_1)$

Danksagung

- Einige Folien entstammen aus der Vorlesung „Sichere Netzerwerkkommunikation“ von Prof. Dr. A. Steffen, Hochschule Rapperswil. An dieser Stelle ein recht herzliches Danke schön an Andreas Steffen.