

Verschlüsselung

Rätsel

- Entschlüssele die folgende Nachricht:

- Wenn entschlüsselt:
Pssst!

- Nachricht:
- JULIUS CAESAR



LWNKWU ECGUCT

Einführung Verschlüsselung

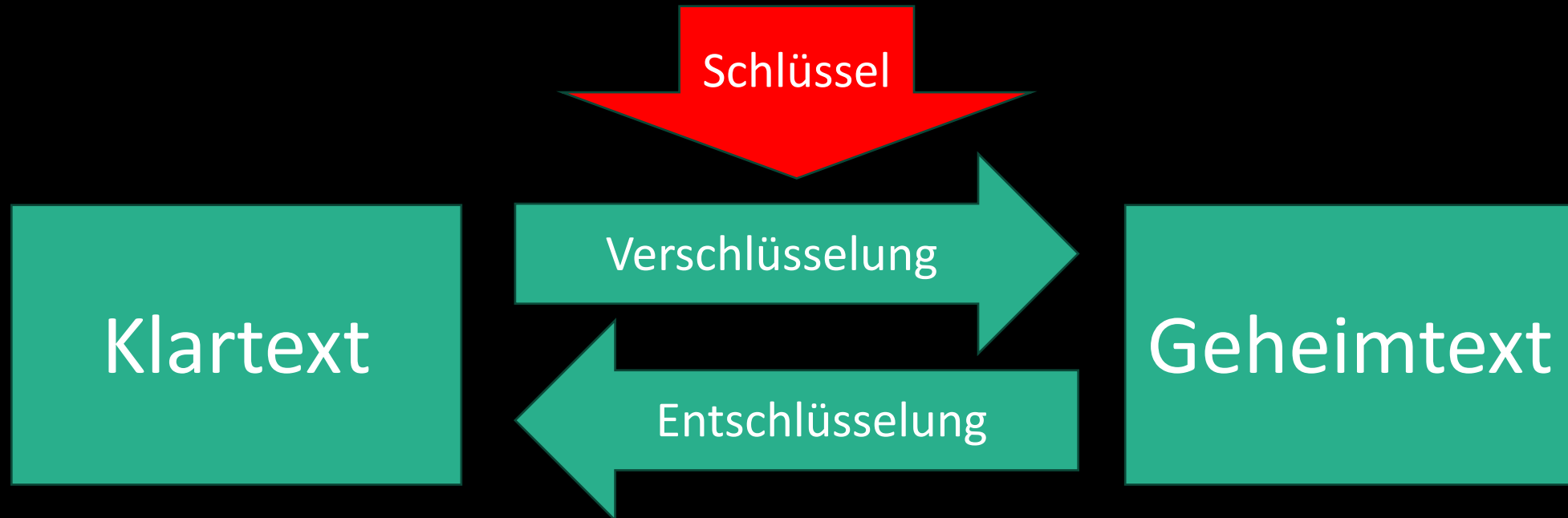
Verschlüsselung

- **Verschlüsselung** oder **Chiffrierung** (engl. encryption):
 - **Umwandlung** von Information, genannt **Klartext** (engl. plaintext, cleartext) ...
 - ... in einen **Geheimtext** / **Chiffre**, (engl. cipher, ciphertext) ...
 - ... mithilfe eines geheimen **Schlüssels** (engl. key).



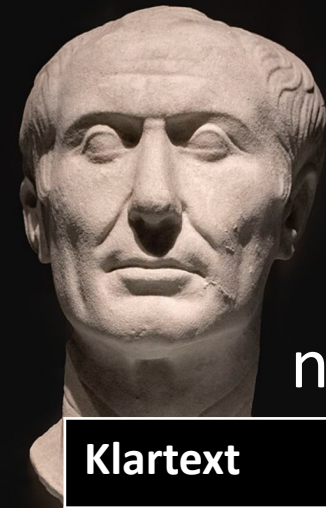
Entschlüsselung

- **Entschlüsseln** (engl. decryption)
 - Gegenteil von Verschlüsseln:
 - Mit geheimen Schlüssel den unlesbaren Geheimtext ...
 - ... in lesbaren Klartext umwandeln



Caesar Verschlüsselung/Chiffre

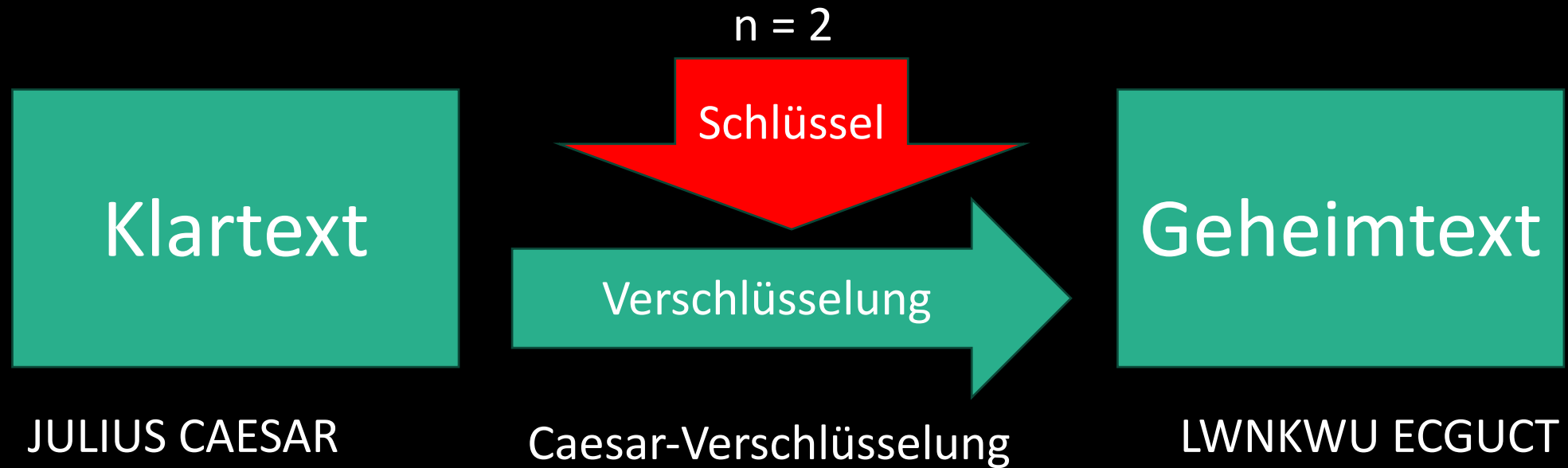
- Jeder Buchstabe um n Stellen verschoben
- Geht auf Gaius Julius Caesar (100-44 vChr.) zurück
- Für militärische Korrespondenz
- Schriftsteller Sueton über Caesar:
„[...] wenn etwas Geheimes zu überbringen war, schrieb er in Zeichen, das heißt, er ordnete die Buchstaben so, dass kein Wort gelesen werden konnte: Um diese zu lesen, tauscht man den vierten Buchstaben, also D für A, aus und ebenso mit den restlichen.“



$n = 2$

Klartext	Geheimtext
A	C
B	D
C	E
...	...
X	Z
Y	A
Z	B

Caesar Verschlüsselung/Chiffre



- Mit Caesar verschlüsselt:
 - Schwierigster Teil ist herauszufinden, welche Art der Verschlüsselung vorliegt
 - Den Schlüssel zu finden ist dann einfach
- Moderne Verschlüsselungsmethoden:
 - Typischerweise ist klar, welcher Verschlüsselungsalgorithmus angewendet wurde
 - Schwierigkeit: Schlüssel herausfinden

Verschlüsselung heute

- Diskussion in Gruppe:
 - Wo wird **heute** verschlüsselt? Warum?
- Mögliche Antworten:
 - Private Kommunikation (E-Mail, WhatsApp, ...)
 - Besuch Internetseiten (z.B. Mail, E-Banking, ...)
 - Kommunikation in Konflikten/Kriegen
 - Geheimdienste
 - Verschlüsselung von Files (z.B. Passwortmanager)
 - Computerfestplatten (unbedingt für Laptop)
 - Kryptowährungen
 - ...



Hauptziele Thema

- Verschiedene **Verschlüsselungsmethoden** kennenlernen
- Einige einfache (wie Caesar) in **Python implementieren**

-> Auftrag Wiki

Vigenère- & Monoalphabetische Verschlüsselung

Caesar-Verschlüsselung (Erinnerung)

- Verschiebe jeden Buchstaben aus Klartext um n Stellen
- Beispiel:
 - Klartext: «**ZAUBERER**» mit $n = 3$
 - Alphabet: «**A****B****C****D****E****F****G****H****I****J****K****L****M****N****O****P****Q****R****S****T****U****V****W****X****Y****Z**»
 - Verschoben: «**D****E****F****G****H****I****J****K****L****M****N****O****P****Q****R****S****T****U****V****W****X****Y****Z****A****B****C**»
 - Geheimtext: «**C****D****X****E****H****U****H****U**»
- Ist offensichtlich, dass keine sehr sichere Verschlüsselungsmethode
- Diskussion:
 - Wie kann man verbessern? Grundidee, dass Buchstaben aus Klartext *verschoben* werden, soll bestehen bleiben.

Vigenère-Verschlüsselung

- Verschlüsselung: **Verschiebung mit Schlüsselwort (key)**
- Beispiel:
 - Klartext: «KANTI ROMANSHORN»
 - Passwort: «ACDC», Verschiebung also um: 1,3,4,3
 - Alphabet: «ABCDEFGHIJKLMNOPQRSTUVWXYZ»
 - Verschlüsselung:
 - Klartext: «KANTI ROMANSHORN»
 - Key: «ACDCACDCACDCACDC»
 - Geheimtext: «LDRWJ VRNDRVIRVQ»

Monoalphabetische Verschlüsselung

- Alphabet wird *durchmischt*
- Beispiel:
 - Klartext: « INFORMATIK »
 - Alphabet: « ABCDEFGHIJKLMNOPQRSTUVWXYZ »
 - Key: « VSEIYHJTBUPANRCOQDLXFWKMZG »
 - Geheimtext: « BRHCDNVXBP »

Aufgaben B

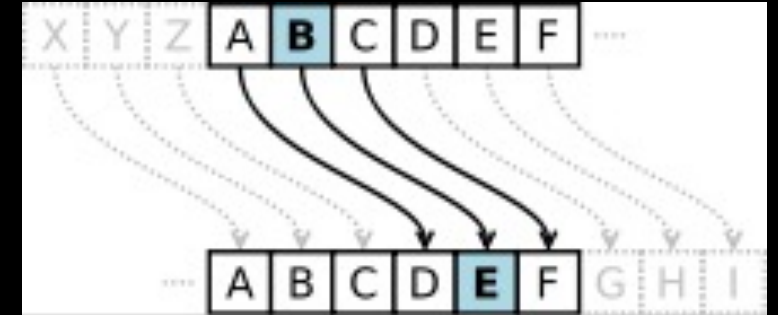
- Ziele:
 - Beide Verschlüsselungsalgorithmen (Vigenère & monoalph.) wirklich verstehen ...
 - ... und von Hand anwenden ...
 - ... und in Python implementieren.
 - Sicherheit dieser beiden Algorithmen analysieren.

Häufigkeitsanalyse

Monoalphabetische Verschlüsselung

- **Caesar Verschlüsselung:**

- Einfachster Fall
- 26 Möglichkeiten (falls nur Grossbuchstaben)
- Extrem einfach zu entschlüsseln: Alle 26 Mögl. durchprobieren



- **Allgemeine Monoalphabetische Verschlüsselung:**

- «ABCDEFGHIJKLMNOPQRSTUVWXYZ» -> «UELXBICNYAZJSQORTPKFMGVHWD»
- Anzahl mögliche Verschlüsselungen:
$$26! \approx 4 \cdot 10^{26}$$
- Vergleich: Masse der Erde $6 \cdot 10^{24}$ kg, Masse Saturn: $6 \cdot 10^{26}$ kg
- Wie **knacken**?
- Brute-Force:
 - Alle möglichen Permutationen durchgehen
 - Zeit mit aktuellen Computern: Millionen bis Milliarden Jahre
 - Kurz: keine gute Idee
- Diskussion: Bessere Ideen?

Häufigkeitsanalyse

- Verschiedene Buchstaben im Alphabet kommen **statistisch unterschiedlich häufig** oft vor in deutscher Sprache
- Rechts: Häufigkeitsanalyse für Goethes Faust
- Ganz klar **häufigster Buchstabe** im Deutsch: E (ca. 16%)
- Beispiel: Verschlüsselter Ausschnitt aus Buch (optionale Aufgabe):

- **Text:** «rx jlw rxn wjxnzvc qr zqtjnbvxivt ljrrvx 4 isxvl nbyzm wsxsjo, tslm jlw tsx lyxrsz mj nvql, nvhx nbyzm nytsx. lqvrslw isxv sjo wqv qwvv tvuyrrvl, nqv uyllbvl nqkh ql vqlv vxuijxwqtv jlw tvhvqrlqngyzzv tvnkhqkbbv gvxbxqkuvl, wvll rqb nyzkhr jlnqll iyzzbvl nqv lqkbbn mj bjl hsevl. rx wjxnzvc isx wqxvubyx vqlvx oqxrs lsrvin txjllqtn, wqv eyhxrskhqlvl hvxbvzzbv. vx isx txynn jlw ejzzqt jlw hsbbsv osnb uvqlvl hszn, wsojx sevx vqlvl nvhx txynnlv nkhjxxesxb. rxn wjxnzvc isx wjll jlw ezylw jlw evnsnn wyddvzb ny gqvz hszn, iqv lybivlwqt tvivnl isxv, isv szvxwqltn nvhx ljbmxqkh isx, wvll ny uyllbv nqv wvl hszn jevx wvl tsxbvlmsjl xvkuvl jlw mj wvl lskhesxl hqljevxdshvl. wqv wjxnzvcn hsbbsv vqlvl uzvqlvl nyhl lsrvin wjwzvc jlw ql qhxl sjtl tse vn lqxtvlwiy vqlvl dxskhbatvxl fjltvl. wqv wjxnzvcn evnsnnvl szzv, isv nqv iyzzbvl, wykh nqv hsbbsv sjkh vql tvhvqrlq, jlw wsnn vn fvrlw sjowvkuvl uyllbv, isx qhxxv txynnbv nyxtv. vqloskh jlvxbstzqkh isxv vn, ivll wqv nskhv rqb wvl dybbvxv hvxsjnuuyrrvl ijxwv. rxn dybbvx isx wqv nkhibnbvx gyl rxn wjxnzvc; wykh wqv evqvv hsbbsv nqkh nkhyv nvqb vbzqkhvl fshxvl lqkbb rvhx tvnvhl. rxn wjxnzvc evhsjdbvbn nytsx, wsnn nqv tsx uvqlv nkhibnbvx hsbbsv, wvll wqvv jlw wvxvl lqkbbnljbm gyl vqlvr rsl isxvl ny jlwjxnzvcshob, iqv rsl vn nqkh lxx wvluvl uyllbv. wqv wjxnzvcn nkhsjwvxv evqr tvwsluvl wsxsl, isv wqv lskhesxl nstvl ijxwvl, nyzzbvl wqv dybbvxv vqlvn bstvn ql qhxxv nbxsnnv sjouxvjmv. wqv wjxnzvcn ijnnbv, wsnn sjkh wqv dybbvxv vqlvl uzvqlvl nyhl hsbbsv, wykh wvl hsbbsv nqv lqv tvnvhl. sjkh wqvvx fjltv isx vql tjbxv txjlw, nqkh gyl wvl dybbvxv ovxlmjhszbsv; rqb vqlvr nyzkhlv uqlw nyzzbv qhx wjwzvc lqkbb ql evxjxjlt uyrrvl.»

- Häufigster Buchstabe: V (15.79%)
- Kann sicher sein, dass das V im Geheimtext dem E im Klartext entspricht
- Weiter für 2. häufigsten Buchstaben usw.
- Irgendwann wird nicht mehr ganz stimmen, dann Buchstaben vertauschen (Trial and Error)
- Ziel: Herausfinden, aus welchem Buch der Text ist.

- a: 4.81%
- b: 1.97%
- c: 4.27%
- d: 5.02%
- e: 16.43%
- f: 1.48%
- g: 2.33%
- h: 6.53%
- i: 8.47%
- j: 0.1%
- k: 1.33%
- l: 3.99%
- m: 3.84%
- n: 9.19%
- o: 2.15%
- p: 0.64%
- q: 0.1%
- r: 6.6%
- s: 6.78%
- t: 5.32%
- u: 5.19%
- v: 0.54%
- w: 1.84%
- x: 0.0%
- y: 0.0%
- z: 1.07%

Vigenère-Verschlüsselung

- Verschlüsselung: **Verschiebung mit Schlüsselwort**
- Beispiel:
 - Schlüsselwort: ABC
 - Klartext: INFORMATIK
 - Verschiebung: ABCABCABCA
 - Geheimtext: JPIPTPBVLL
- Ist *keine* monoalphabetische Verschlüsselung, da bestimmter Buchstabe im Klartext nicht immer mit gleichem Buchstaben verschlüsselt wird. Ist eine **polyalphabetische Verschlüsselung**.

Vigenère-Verschlüsselung

- Ist es möglich, damit verschlüsselte Nachricht zu knacken?
- Ja
- Zumindest wenn Nachricht deutlich länger ist als Schlüsselwort.
- Nicht knackbar:
 - Nachricht: «INFORMATIK»
 - Schlüsselwort: «KANTIROMANSHORNDIEINNOVATIVESCHULEIMGRUENEN»
- Beispiel knackbar:
 - Nachricht: Goethes Faust (das ganze Buch)
 - Schlüsselwort: «PIZZA»
- Kasiski-Test

Kasiski-Test

- Zum knacken von Nachricht, mit Vigenère verschlüsselt
- Vorgehen:
 - Suche im Text nach Zeichenfolgen, die mehrfach vorkommen ...
 - ... und bestimmte Abstände dazwischen
 - ggT der Abstände ist vielfaches der Länge des Schlüsselworts
 - Beruht darauf, dass gewisse Silben und Wortendungen oft vorkommen in Sprachen

Kasiski-Test

- Geheimtext, mit Vigenère verschlüsselt
- Resultat Kasiski-Test ->
- Häufigster Teiler: 9
- Vermutung: Schlüsselwort ist 9 Zeichen lang
- (stimmt tatsächlich)
- Kann nun Häufigkeitsanalyse von jedem 9. Buchstaben machen

KIQIGIQEOBZTLHCUMHXZAOLKZWNRZBWHZV-
GYAVZSHWNGUYCQGKLVIDJJMRWCQEOELZF-
HXZWLSUCRJDWFLWEMVNIGRATRVEOPAGIJ-
ZIRGVZUHQJKLDITGZGRJFDXMFLWJMMQVM-
TRJEVXQZAULGZADXJMKUJDBSGXMJLNDPMHCB

	Abstand	Teiler		Abstand	Teiler
FLW	45	3, 5, 9, 15, 45	QE	54	2, 3, 6, 9, 18, 27, 54
HXZ	50	2, 5, 10, 25, 50	RJ	43	43
QEO	54	2, 3, 6, 9, 18, 27, 54	RJ	59	59
CQ	13	13	RJ	16	2, 4, 8, 16
DX	26	2, 13, 26	TR	43	43
EO	54	2, 3, 6, 9, 18, 27, 54	VZ	67	67
EO	86	2, 43, 86	WN	16	2, 4, 8, 16
EO	32	2, 4, 8, 16, 32	XM	36	2, 3, 4, 6, 9, 12, 18, 36
FL	45	3, 5, 9, 15, 45	XZ	50	2, 5, 10, 25, 50
GI	93	3, 31, 93	ZA	121	11, 121
GR	30	2, 3, 5, 6, 10, 15, 30	ZA	126	2, 3, 6, 7, 9, 14, 18, 21, 42, 63, 126
GZ	29	29	ZA	5	1, 5
HC	153	3, 9, 17, 51, 153	ZW	45	3, 5, 9, 15, 45
HX	50	2, 5, 10, 25, 50			
IG	83	83			
IQ	4	2, 4			
JD	77	7, 11, 77			
JM	72	2, 3, 4, 6, 8, 9, 12, 18, 24, 36, 72			
JM	94	2, 47, 94			
JM	22	2, 11, 22			
KL	61	61			
LW	45	3, 5, 9, 15, 45			
MH	149	149			

Auftrag

- Siehe Wiki

Vigenère-Game

Vigenère-Game: Auftrag

- In **Klassenchat** miteinander kommunizieren ...
- ... aber **nur mit verschlüsselten Nachrichten** (kein Klartext!!!)
- Mit Vigenère verschlüsselt (Code wird bereitgestellt)
- Jede Person erhält **Schlüsselwort** —> Lesen, eintippen, vernichten!
- Auftrag:
 1. Finde andere **Gruppenmitglieder** (3-4, gleiches Schlüsselwort). Versuche also Nachrichten der anderen Personen mit deinem Schlüsselwort zu entschlüsseln
 2. Löst und diskutiert gemeinsam **Rätsel** (siehe nächste Slides).
- Wichtigste **Regeln**:
 - **Kommunikation nur im Klassenchat** auf Teams (nicht sprechen, keine privaten Chats, keine Telepathie, ...) und ...
 - ... nur **verschlüsselt**.
 - Wenn möglich Heavy Metal (oder anderen Krach) hören mit Kopfhörer!
 - Keine Smileys o.ä.
 - Erst mit Rätsel beginnen, wenn **Gruppe komplett**.

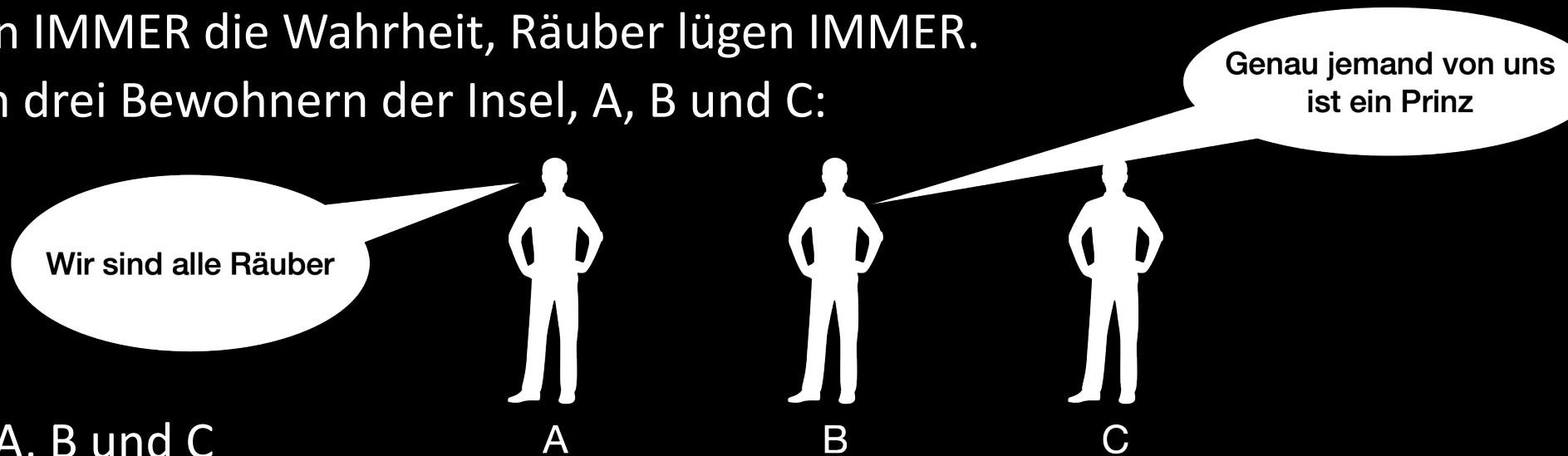
Vigenère-Game: Rätsel

Auftrag:

1. Finde andere **Gruppenmitglieder** (3-4, gleiches Schlüsselwort). Versuche also Nachrichten der anderen Personen mit deinem Schlüsselwort zu entschlüsseln
2. Gemeinsam **Rätsel** lösen

• Rätsel 1:

- Auf der Insel befinden sich ausschliesslich Prinzen und Räuber.
- Prinzen sagen **IMMER** die Wahrheit, Räuber lügen **IMMER**.
- Sie begegnen drei Bewohnern der Insel, A, B und C:



- Identifiziere A, B und C
- Rätsel 2: auf der gleichen Insel begegnest du drei anderen Personen A,B,C
 - A sagt: «B und C» sind von der gleichen Art.
 - Du fragst C: «Sind A und B von der gleichen Art?»
 - Was sagt C?

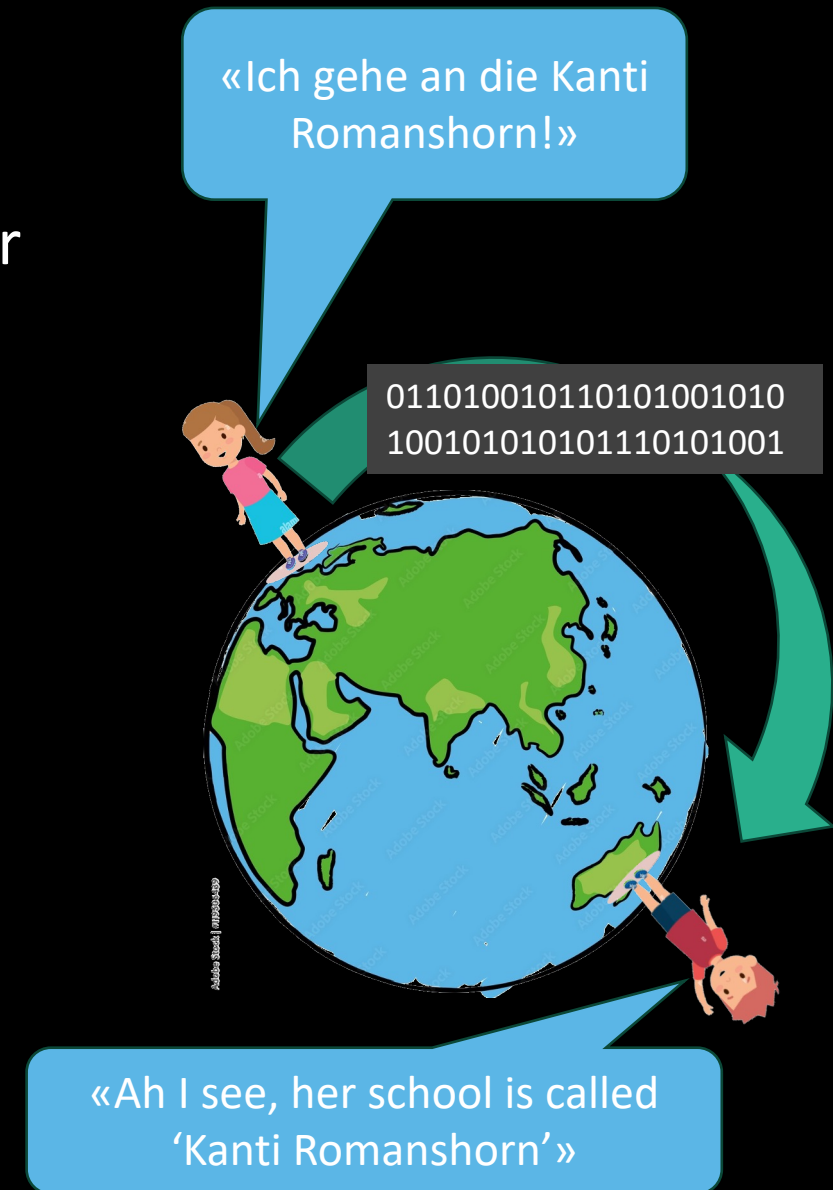
Zeichencodierung I

Zeichencodierung

- Bisherige Verschlüsselungsverfahren: haben gewissermassen Buchstaben in Zahlen umgewandelt
- Beispiel: Vigenère-Verschlüsselung
 - Klartext: «INFORMATIK» mit Schlüsselwort «ABC»
 - Erster Buchstabe (I) wird um 1 Stelle (A) verschoben, zweiter um 2 Stellen, ...
- Jede Zahl steht also für ein Symbol:
 - 1 für A, 2 für B, 3 für C, ...
- Nennt eine *Zuordnung Zahl zu Symbol* ein **Zeichencodierung**
- Wollen uns jetzt mit Zeichencodierungen, wie sie in Computern verwendet werden, auseinandersetzen
- Beachte: Zeichencodierung per se ist *keine* Verschlüsselung.

Zeichencodierung

- Möchte Nachricht «Ich gehe an die Kanti Romanshorn!» meinem Freund Jack in Australien per Computer (und unverschlüsselt) übermitteln.
- Computer kennt auf unterster Ebene aber nur Binärsystem (0 und 1).
- Nachricht muss also in 0 und 1 umgewandelt werden (encoding) ...
- ... diese werden nach Australien übermittelt ...
- ... und dort in lesbaren Text zurückübersetzt (decoding) werden.
- Damit klappt, müssen Jack und ich uns auf eine *Zeichencodierung einigen*. Verwenden wir unterschiedliche Zeichencodierungen, erhält Jack Kauderwelsch.



ASCII

- 1963 wurde ASCII-Zeichentabelle eingeführt
- «**American Standard Code for Information Interchange**»
- 7-Bit Zeichencodierung
- Probleme mit ASCII?
- Viele Zeichen fehlen:
 - Umlaute ä,ö,ü
 - Zeichen anderer Sprachen (z.B. Chinesisch)
 - Keine 😊 😄 😁 😂 😃 😅 😆 😇

0	NUL	16	DLE	32		48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	\$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

- Ersten 32 Zeichen sind **Steuerzeichen** (control characters):
 - SOH: Start of Heading (Beginn der Kopfzeile)
 - LF: Line Feed (Zeilenvorschub): Zeilenumbruch
 - <https://de.wikipedia.org/wiki/Steuerzeichen>

ASCII

0 NUL	16 DLE	32	48 0	64 @	80 P	96 `	112 p
1 SOH	17 DC1	33 !	49 1	65 A	81 Q	97 a	113 q
2 STX	18 DC2	34 "	50 2	66 B	82 R	98 b	114 r
3 ETX	19 DC3	35 #	51 3	67 C	83 S	99 c	115 s
4 EOT	20 DC4	36 \$	52 4	68 D	84 T	100 d	116 t
5 ENQ	21 NAK	37 %	53 5	69 E	85 U	101 e	117 u
6 ACK	22 SYN	38 &	54 6	70 F	86 V	102 f	118 v
7 BEL	23 ETB	39 '	55 7	71 G	87 W	103 g	119 w
8 BS	24 CAN	40 (	56 8	72 H	88 X	104 h	120 x
9 HT	25 EM	41)	57 9	73 I	89 Y	105 i	121 y
10 LF	26 SUB	42 *	58 :	74 J	90 Z	106 j	122 z
11 VT	27 ESC	43 +	59 ;	75 K	91 [	107 k	123 {
12 FF	28 FS	44 ,	60 <	76 L	92 \	108 l	124
13 CR	29 GS	45 -	61 =	77 M	93]	109 m	125 }
14 SO	30 RS	46 .	62 >	78 N	94 ^	110 n	126 ~
15 SI	31 US	47 /	63 ?	79 O	95 _	111 o	127 DEL

- Nachricht: «Ich gehe an die Kanti Romanshorn!»

- ASCII (dezimal):

```
73 99 104 32 103 101 104 101 32 97 110 32 100 105 101 32 75
97 110 116 105 32 82 111 109 97 110 115 104 111 114 110 33
```

- ASCII (binär):

```
1001001 1100011 1101000 0100000 1100111 1100101 1101000 1100101 0100000
1100001 1101110 0100000 1100100 1101001 1100101 0100000 1001011 1100001
1101110 1110100 1101001 0100000 1010010 1101111 1101101 1100001 1101110
1110011 1101000 1101111 1110010 1101110 0100001
```

- Nachricht kann nun im ASCII-Binärcode übermittelt werden ...
- ... und am Zielort mithilfe der ASCII-Zeichentabelle zurückübersetzt werden

ASCII mit Python

- ASCII-Code von Zeichen bestimmen:
`ord('t') # 116`
- Zeichen von ASCII-Code bestimmen:
`chr(116) # 't'`

0	NUL	16	DLE	32		48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	\$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Aufgaben

- Aufgaben D1-D3
- Optional: D4

Zeichencodierung II

Zeichentabellen

- ASCII alleine reicht also nicht aus
- Keine Emojis 🤖
- Deshalb benötigte viele weiteren Zeichentabellen für weitere Zeichen
- -> umständlich & fehleranfällig
- Ausweg: **Unicode**

Unicode

- 1991: Veröffentlichung Version 1.0.0
- Ist universeller Zeichensatz
- Jedes Zeichen besteht aus 1-4 Bytes (1 Byte = 8 Bit) ...

- ... und hat Form:

Länge	Byte 1	Byte 2	Byte 3	Byte 4	Anzahl Zeichen
1 Byte	0??? ????				
2 Byte	110? ????	10?? ????			
3 Byte	1110 ????	10?? ????	10?? ????		
4 Byte	1111 0???	10?? ????	10?? ????	10?? ????	

- Ist **präfixfreier Code**

Präfixfreier Code

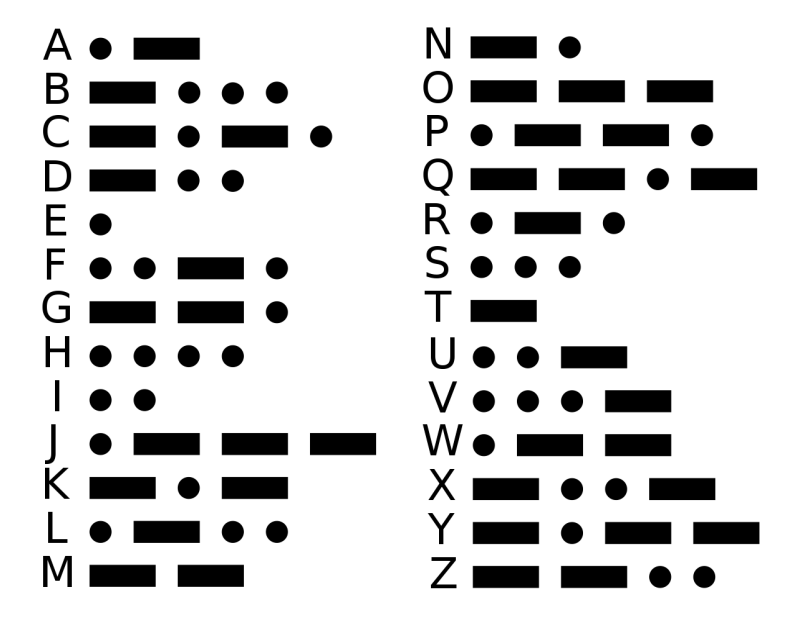
- **Präfixfreier Code:**

- Code mit variabler Zeichenlänge, ...
- ... bei dem immer klar ist, wo neues Zeichen beginnt.
- Es ist also kein Präfix nötig.

- **Beispiel nicht präfixfreier Code: Morzen**

- Was bedeutet: * * * * - - ?
- Mehrere Möglichkeiten:
 - HM
 - EEEEM
 - IEW
 - ...

- Damit eindeutig ist, benötigt *Präfix* (z.B. Abstand, ...) am Anfang von jedem Symbol:
 - pause**pause*pause*--



A	• —	N	— •
B	— • • •	O	— — —
C	— • — •	P	• — — •
D	— • •	Q	— — • —
E	•	R	• — •
F	• • — •	S	• • •
G	— — •	T	—
H	• • • •	U	• • —
I	• •	V	• • • —
J	• — — —	W	• — —
K	— • —	X	— • • —
L	• — • •	Y	— • — —
M	— —	Z	— — • •

Unicode



Länge	Byte 1	Byte 2	Byte 3	Byte 4
1 Byte	0??? ????			
2 Byte	110? ????	10?? ????		
3 Byte	1110 ????	10?? ????	10?? ????	
4 Byte	1111 0???	10?? ????	10?? ????	10?? ????

- Ist präfixfreier Code
- Immer klar, wo neues Zeichen beginnt.
- Beispiel:

11010011 10100111 01101010 11110011 10011010 10010110 10100101

Zeichen 1

Zeichen 2

Zeichen 3

- Stand 2024: Ca. 150'000 Zeichen in Unicode
- Unicode Konsortium (Non-Profit Organisation) entscheidet über Aufnahme neuer Zeichen

Unicode



- Gibt verschiedene Kodierungen von Unicode, am bekanntesten ist ...
- UTF-8
 - UCS Transformation Format, wobei UCS: Universal Coded Character Set
 - Stimmt in ersten 128 Zeichen mit ASCII überein
 - Stand 2024: Ca. 98% aller Websites verwenden UTF-8
- Python:
 - `ord( )` und `chr( )` Funktionen von vorher funktionieren für erste 256 Zeichen von UTF-8

Auftrag

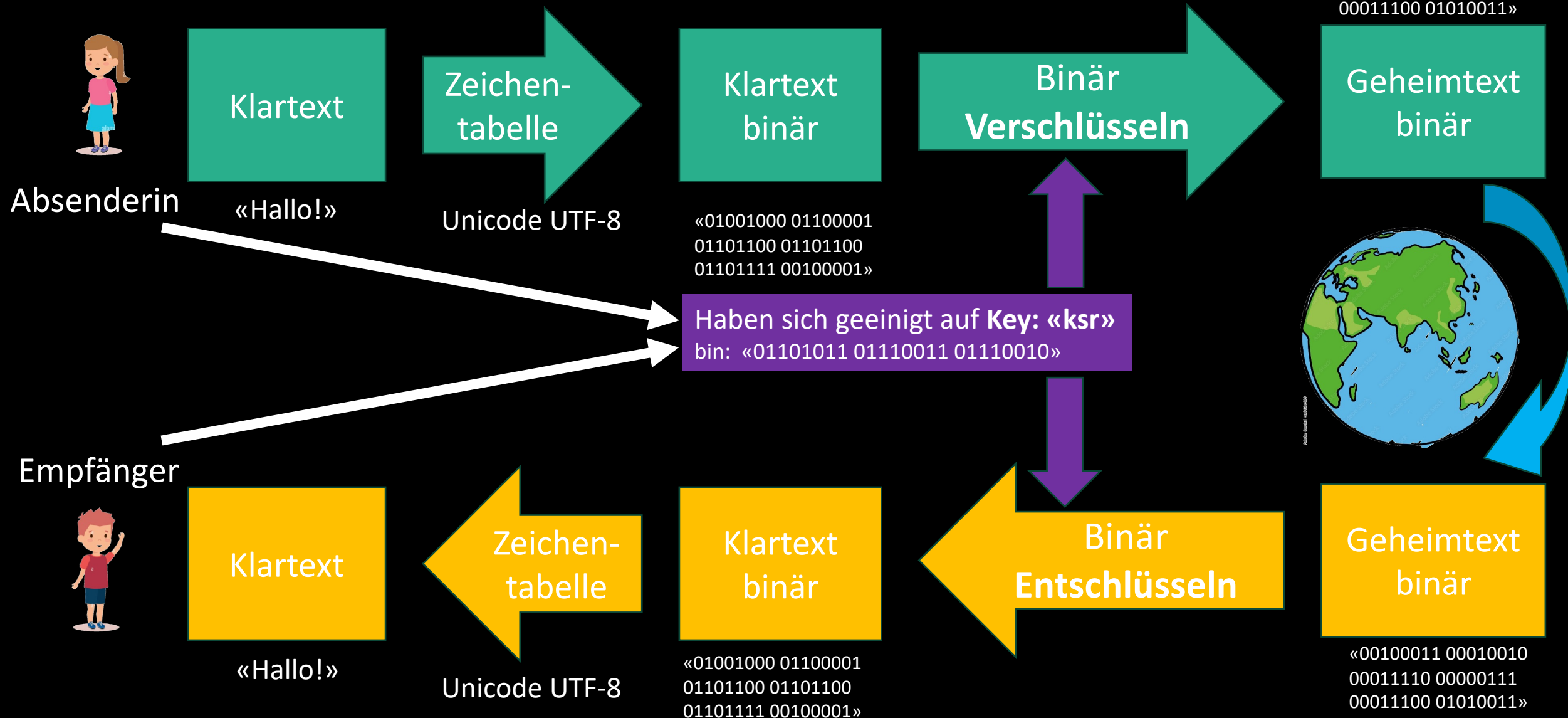
- Restliche Aufgaben D

XOR-Verschlüsselung

Verschlüsselung von Binärzahlen

- **Zeichencodierung / Zeichentabelle:**
Zuweisung: Zahl $\leftrightarrow$ Zeichen
- Damit können Zeichen in Binärzahl umwandeln
- Beispiel 'A' $\rightarrow$ 65 $\rightarrow$ 1000001
- Wollen nun **auf Ebene von Binärzahlen Nachricht verschlüsseln**

Verschlüsselung von Binärzahlen



Verschlüsselung von Binärzahlen

- Doch wie Binärzahlen verschlüsseln?
- Bisherige Verschlüsselungsmethoden funktionieren nicht, da 'Alphabet' nur aus zwei Zeichen besteht: 0 und 1
- Ausweg: XOR-Verschlüsselung

Logik-Operatoren: AND, OR und XOR

- UND (**AND**) in Logik/Informatik:
 - Aussage «A AND B» ist nur wahr, wenn *beide* Aussagen A, B wahr sind
 - Python: `if x > 0 and x < 10: ...`
- ‘normales’ ODER (**OR**):
 - Aussage «A OR B» ist wahr, wenn mind. eine der beiden Aussagen A, B wahr ist
 - Python: `if x > 0 or x < 10: ...`
- eXclusive OR (**XOR**):
 - Aussage «A XOR B» ist wahr, wenn *genau eine* der beiden Aussagen wahr ist
 - Python: später

Wahrheitstabellen:

(Wahr = 1, Falsch = 0):

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

XOR-Verschlüsselung

- Speziell an XOR: Wendet man es **2x** an, kommt man wieder an **Ursprung**:
 $(A \text{ XOR } B) \text{ XOR } B = A$
- Nutzen nun XOR, um Binärzahl bitweise zu verschlüsseln:
 - zwei Bits so verknüpft, dass das Resultat genau dann 1 ist, wenn der eine oder der andere der Operanden 1 ist, aber nicht beide
 - Notation: p für Plaintext/Klartext, k für Key, c für Ciphertext/Geheimtext

Verschlüsselung:

p	k	c = p XOR k
0	0	0
0	1	1
1	0	1
1	1	0

Entschlüsselung:

c	k	p = c XOR k
0	0	0
0	1	1
1	0	1
1	1	0

Beispiele

- Verwendet oft **^-Operator** für XOR

- Beispiele Binärzahlen:

- $0110 \wedge 1100 = \dots$ **1010**
- $10010101 \wedge 11111100 =$ **01101001**
- $1101 \wedge 1101 =$ **0000**
- $11011 \wedge 110 =$ **11101**

- Beispiele Dezimalzahlen:

- $3 \wedge 4 =$ **7, weil: $3 \wedge 4 = 11_2 \wedge 100_2 = 111_2 = 7$**
- $6 \wedge 5 =$ **3**
- $42 \wedge 42 =$ **0**
- $6 \wedge 12 =$ **10 (1. Beispiel)**

XOR mit Python

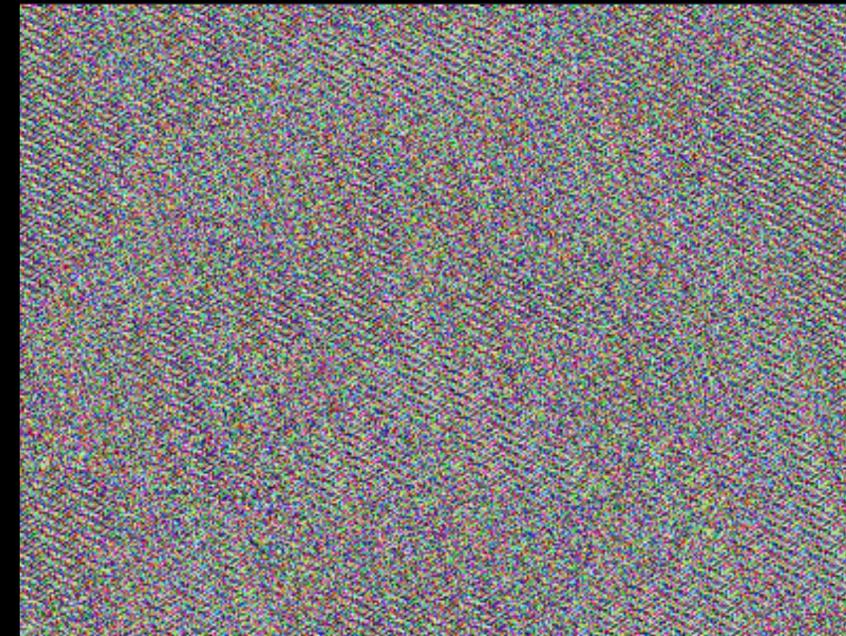
- Kann einfach XOR direkt auf Dezimalzahlen anwenden
- `print(3^4) # = 7`
- Muss also nicht Zahl zuerst in Binärzahl umwandeln, das macht Python im Hintergrund für uns
- Weitere Python-Tipps:
 - Binärzahl -> Dezimalzahl: `int("0110", 2)`
 - Dezimalzahl -> Binärzahl: `bin(65)[2:]`

Warum XOR?

- Da XOR-Verschlüsselung auf Ebene von Binärzahlen agiert ...
- ... können **alles damit verschlüsseln**, was als Binärzahl gespeichert wird
- Also: Alle Files auf Computer!
- Z.B. Bilder!

Bild-Verschlüsselung mit XOR

- Bild: 2x mit XOR verschlüsselt, keys:
 - Key1 = 10010011001111111
 - Key2 = 110111010000010011111111011000001110



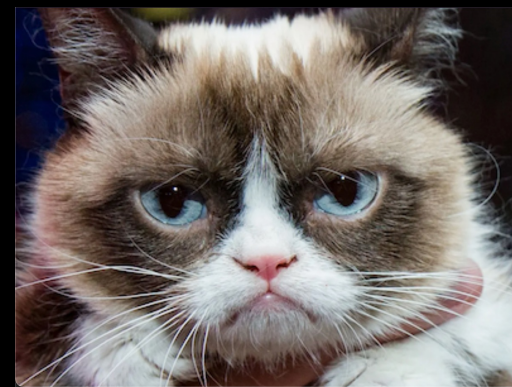


Bild-Verschlüsselung mit XOR

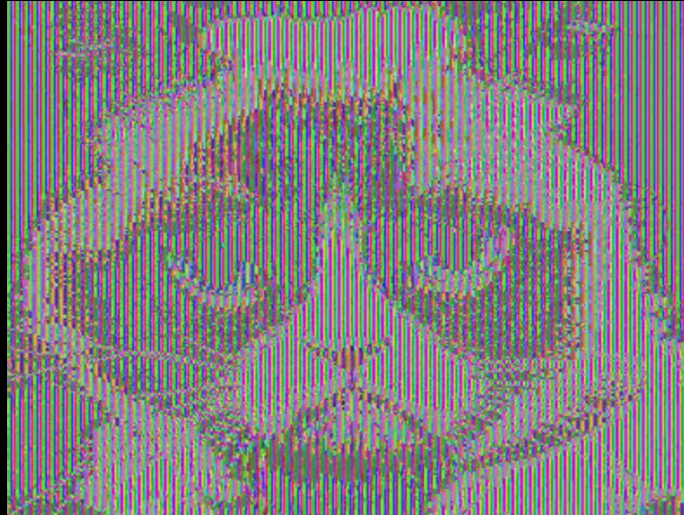
- Schwierigkeit: Bildinformation besteht aus sehr vielen Bits
- Kleines Bild Grumpy Cat: 2'880'000 Bits
- Ist Schlüssel deutlich kürzer als dies -> Wiederholungen -> Erkennbar
- Ausweg:
 1. Sehr langer Schlüssel
 2. Bild mehrfach hintereinander verschlüsseln
- Nächste Slide: gleiches Bild mit Schlüsseln aus versch. Anzahl Bits

Bild-Verschlüsselung mit XOR

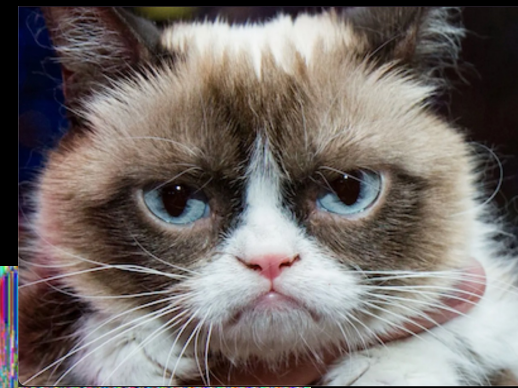
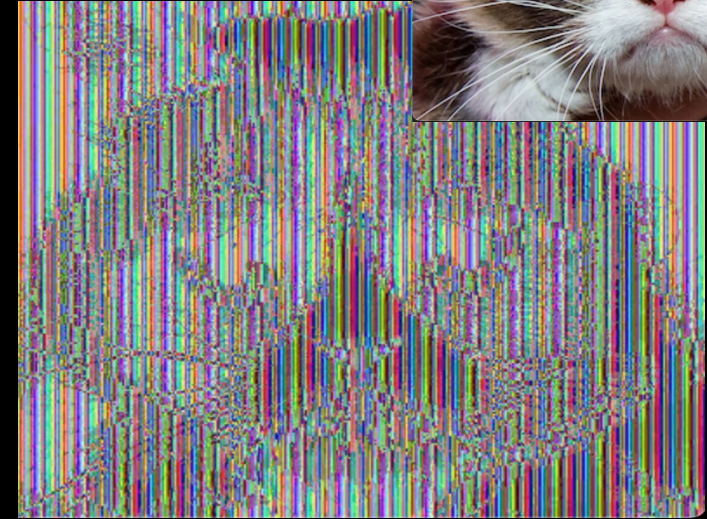
2 Bits



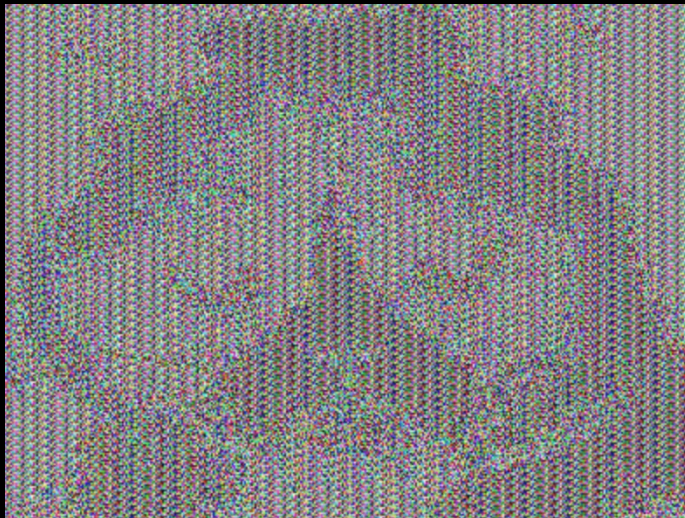
10 Bits



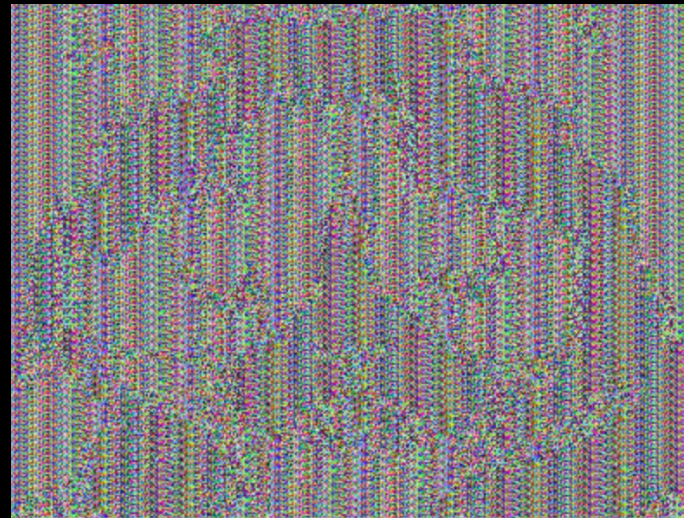
100 Bits



1000 Bits



2000 Bits



100'000 Bits

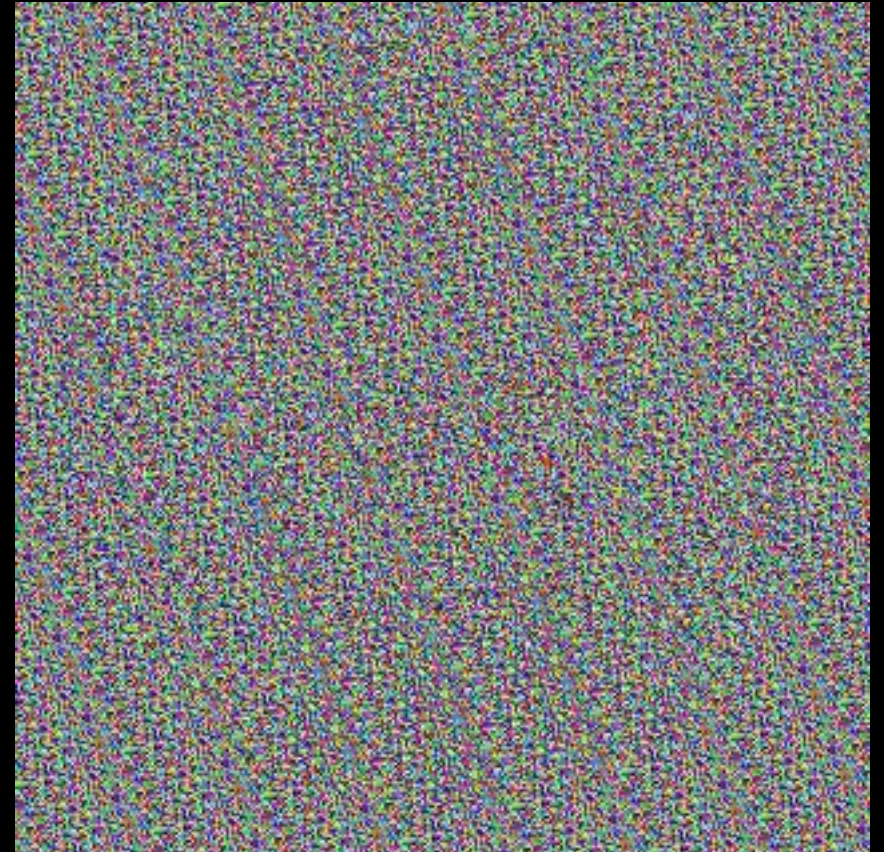


Bild-Verschlüsselung mit XOR

- Zusatzaufgabe: Entschlüssele das folgende Bild (siehe Wiki):

Schlüssel:

```
"111100000110011011000100001101011100101110011001000101100001111100011001010001  
011011101111110000000000110000001111100101111010101111100011101010101010000011  
11001010110110100111000000000011111110100011111001101111010011110100000010111  
010100001011111101010011000101010010110111011101010110011101001000011011101100  
110001101001001111000100110011001001110000000000110111100001000100011101011  
001111010101001111000100010101001011001111011010111001101100000000001010000000  
101001111100100000111010001110011000001010111100000001001100110000010001010010  
0001110100010010010010001010011111001110101101010101100010110100010010100001  
1100010010001011100111110001110000100001100110001110011010110111110010001100010  
01111010111111100001111111111110111011001011111000001010001011001010111111111  
100010101110001010101100110110101111111011111101000001001001010010001001011001  
111000101010111111110010011010000110011011110001111111101011000011001101100001  
011000000000101111100010110110100100100001110100100110011010100101000000111011  
111101001010011111100110100001010011110011110000000101011110010001011001100101  
111011110010000101101111010001101101010001011100010010111001111100101110010111  
111110100000100110000111011001101110010010001100011101100110110000110010100011  
110001011001000010101100110000010111000010110100110100011101010100011110000001  
11111101110011000010010000111010111000111000110"
```



Auftrag

- Siehe Wiki

Moderne Verschlüsselungsmethoden

Symmetrische & Asymmetrische Verschlüsselung

- **Symmetrische Verschlüsselung:** *gleichen Key* zum verschlüsseln und entschlüsseln
- **Asymmetrische Verschlüsselung:** *unterschiedliche Keys* dazu
- Beispiele symmetrische & asymmetrische Verschlüsselung?
 - Symmetrisch:
 - alle bisherigen
 - Caesar
 - Monoalphabetische V.
 - Vigenère
 - XOR
 - Asymmetrisch:
 - keine der bisherigen
 - RSA

Advanced Encryption Standard (AES)

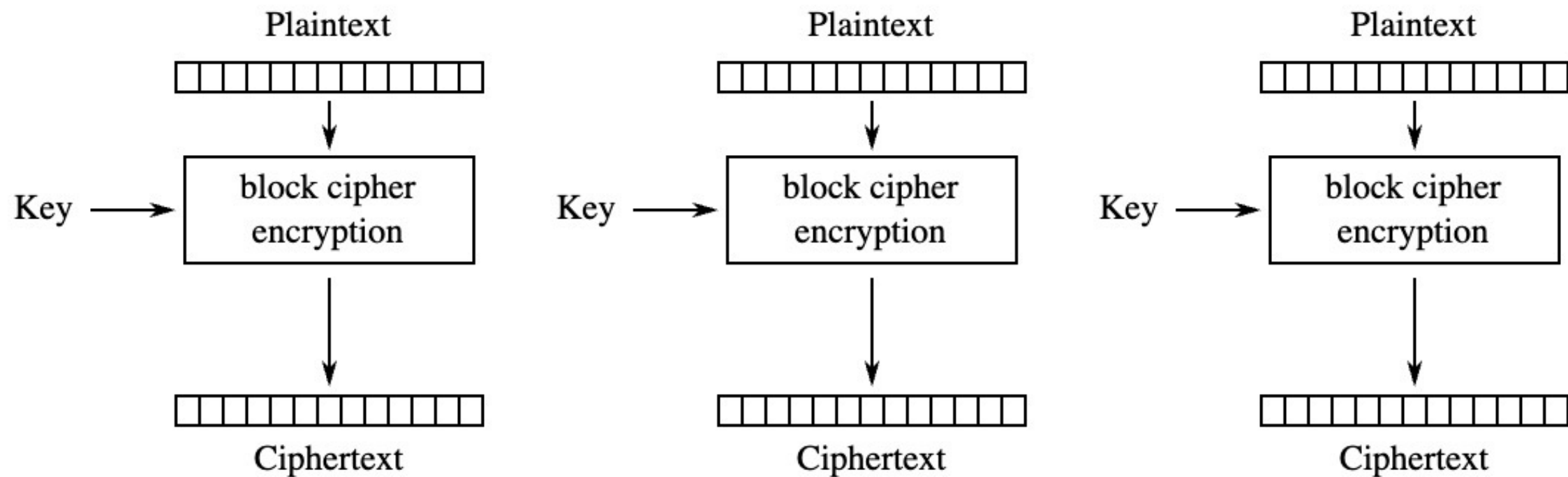
- **AES: Advanced Encryption Standard**
- *Modernes, symmetrisches Verschlüsselungsverfahren*
- Am weitesten verbreitetes symmetrisches Verschlüsselungsverfahren
- Wurde 1997 vom NIST (**N**ational **I**nstitute of **S**tandard and **T**echnology, amerikanisch) aus mehreren Kandidaten ausgewählt und verbreitet
- AES auf modernen Computerprozessoren direkt implementiert
-> sehr schnell

Advanced Encryption Standard (AES)

- Wird nicht einzelnes Zeichen, sondern ein **Block verschlüsselt** (block cipher)
- Blocklänge ist 128 bit
- 128 bit Klartext -> 128 bit Ciphertext
- Verwendet dazu Schlüssel (128, 192 oder 256 bit)
- Nennt AES-128, AES-192, AES-256
- Nennt Verschlüsselung «Block Cipher Encryption»
- Verschlüsselt in mehreren Runden

Advanced Encryption Standard (AES)

- Eine **Runde**:
 1. **Monoalphabetische** Verschiebungen
 2. **Permutationen** (Vertauschen von Positionen)
 3. Bilden von **multiplikativen Inversen** bei modularer Multiplikation:
 - $\frac{1}{7}$ ist Inverses von 7 bei regulärer Multiplikation weil $\frac{1}{7} \cdot 7 = 1$
 - Bsp. *modulare* Multiplikation: $(2 \cdot 4) \% 5 = 8 \% 5 = 3$
 - Zahl 3 ist das Inverse von 2 bei Modulo 5, weil $(2 \cdot 3) \% 5 = 6 \% 5 = 1$
 4. Bitweises **XOR**
- Beachte: für jede Runde wird ein neuer **Roundkey** erstellt: 'Modifikation' vom eigentlichen Key
- Anzahl Runden:
 - 128 bit Key: 10 Runden
 - 192 bit Key: 12 Runden
 - 256 bit Key: 14 Runden

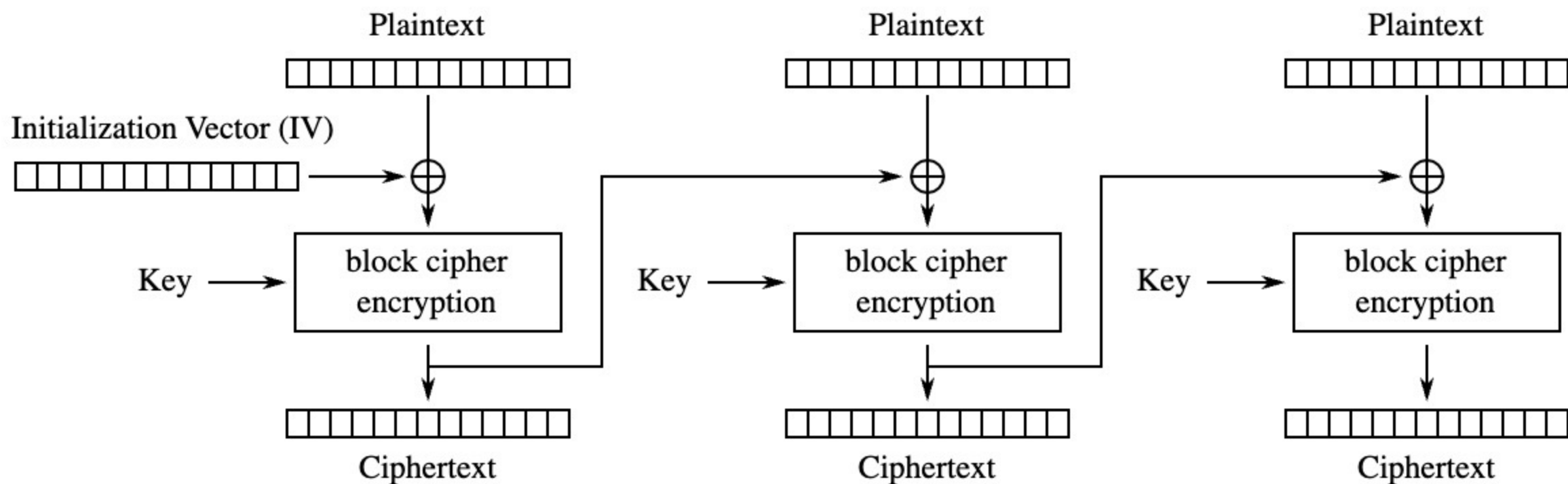


Electronic Codebook (ECB) mode encryption

Verschlüsselung im ECB-Modus

Advanced Encryption Standard (AES)

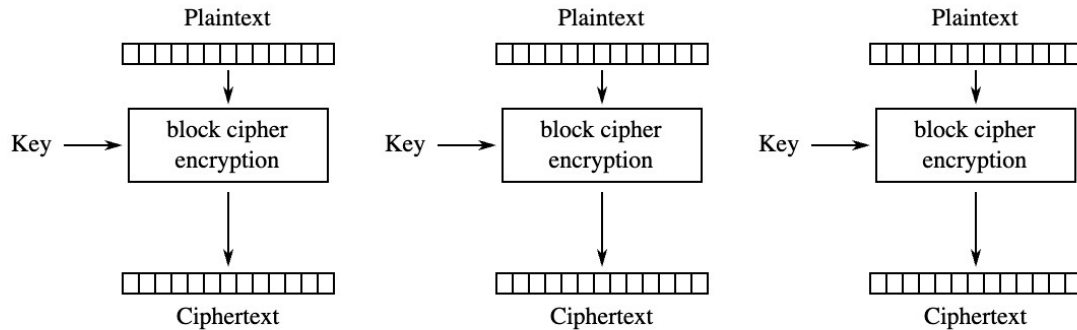
- Entschlüsseln: Schritte von Verschlüsselung rückwärts auf jeden Block anwenden
- Verschlüsselung bisher noch nicht sicher genug.
- Warum?
- Gleiches Problem wie bei Vigenère oder XOR: Wiederholung nach Schlüssellänge
- Lösung: Verschlüsselte Blöcke zusammen verschlüsseln:
 - Ähnlich wie Autokey (war eine Zusatzaufgabe)
 - Idee: Block verwenden, um nachfolgenden Block zu verschlüsseln
 - Heisst **Block Chaining**
 - Ersten Block wird mit **Initialisierungsvektor** verschlüsselt



Cipher Block Chaining (CBC) mode encryption

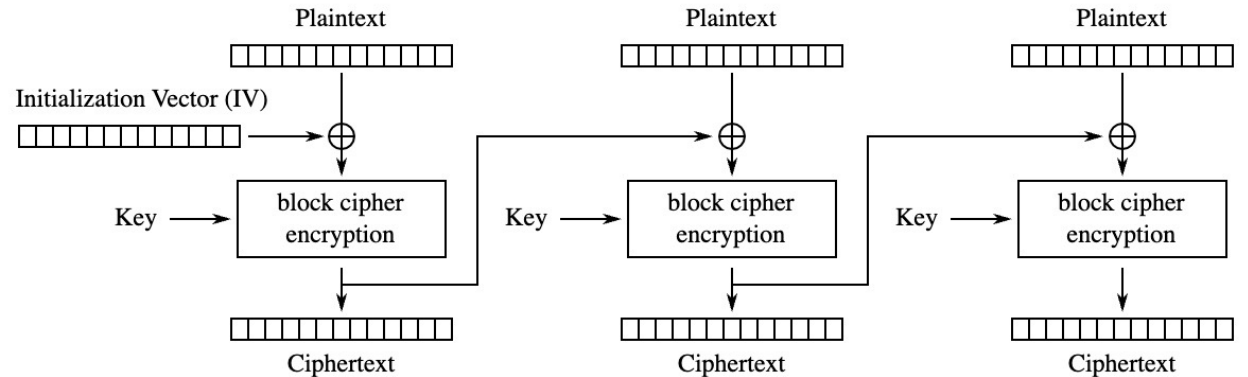
Verschlüsselung im CBC-Modus

AES mit ECB & CBS



Electronic Codebook (ECB) mode encryption

Verschlüsselung im ECB-Modus



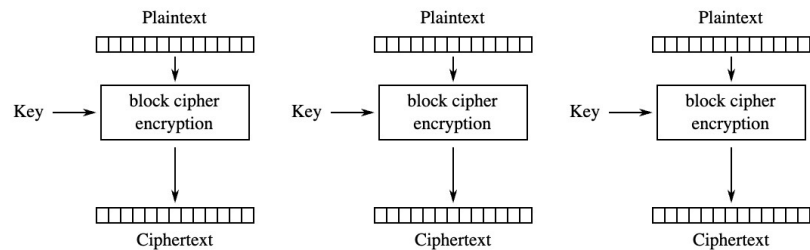
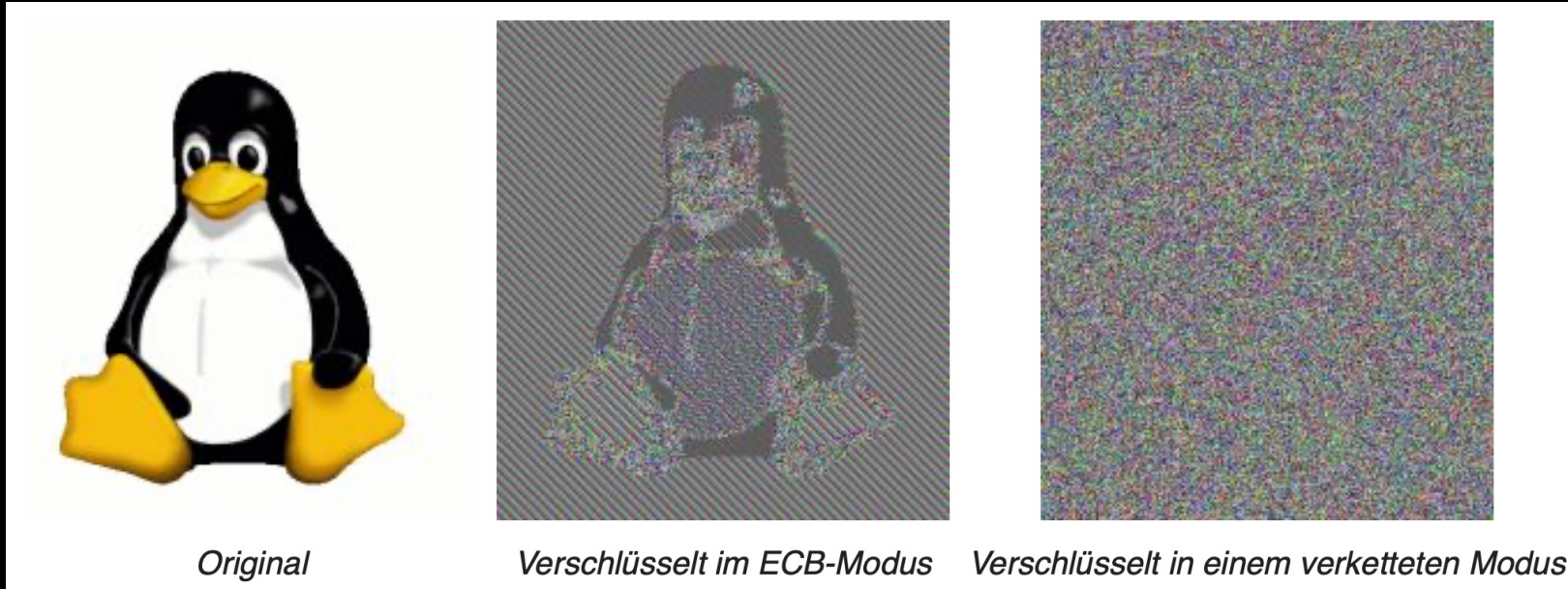
Cipher Block Chaining (CBC) mode encryption

Verschlüsselung im CBC-Modus

- **AES ohne Block Chaining**
- Benötigt nur Key (128, 192 oder 256 bit)

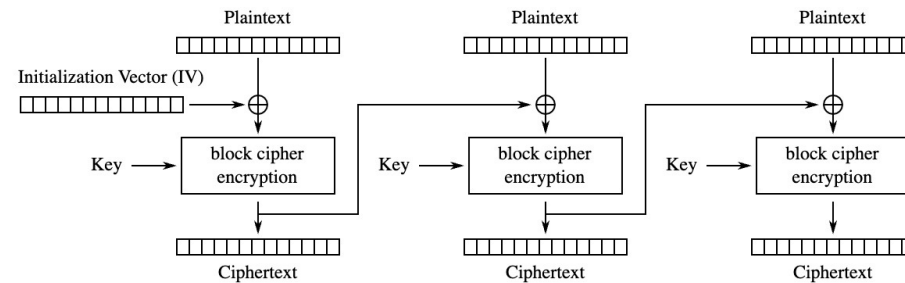
- **AES mit Block Chaining**
- Benötigt Key ...
- ... und Initialization Vector

AES mit ECB & CBS



Electronic Codebook (ECB) mode encryption

Verschlüsselung im ECB-Modus



Cipher Block Chaining (CBC) mode encryption

Verschlüsselung im CBC-Modus

Knacken von AES

- Gilt momentan als nicht knackbar!
- Heutige Computer bräuchten Milliarden von Jahren (oder noch viel mehr), um den richtigen Schlüssel mit Brute Force zu ermitteln
- Zukunft?
- Kann sich ändern mit **Quantencomputern**
- Neue Generation von Computern, an denen aktuell geforscht wird ...
- ... die besonders schnell sind im knacken von Verschlüsselungen sind
- Aktuelles Problem: **SNDL**
- «**Store Now, Decrypt Later**»
- Speichern (stehlen?) von verschlüsselten Daten in Hoffnung, das in Zukunft mit Quantencomputern entschlüsseln kann.

Video

- Video dazu von Veritasium

