Informationssicherheit:
Kryptographie

Teil 1: Geschichte, Symmetrische Kryptographie, Hash-Funktionen

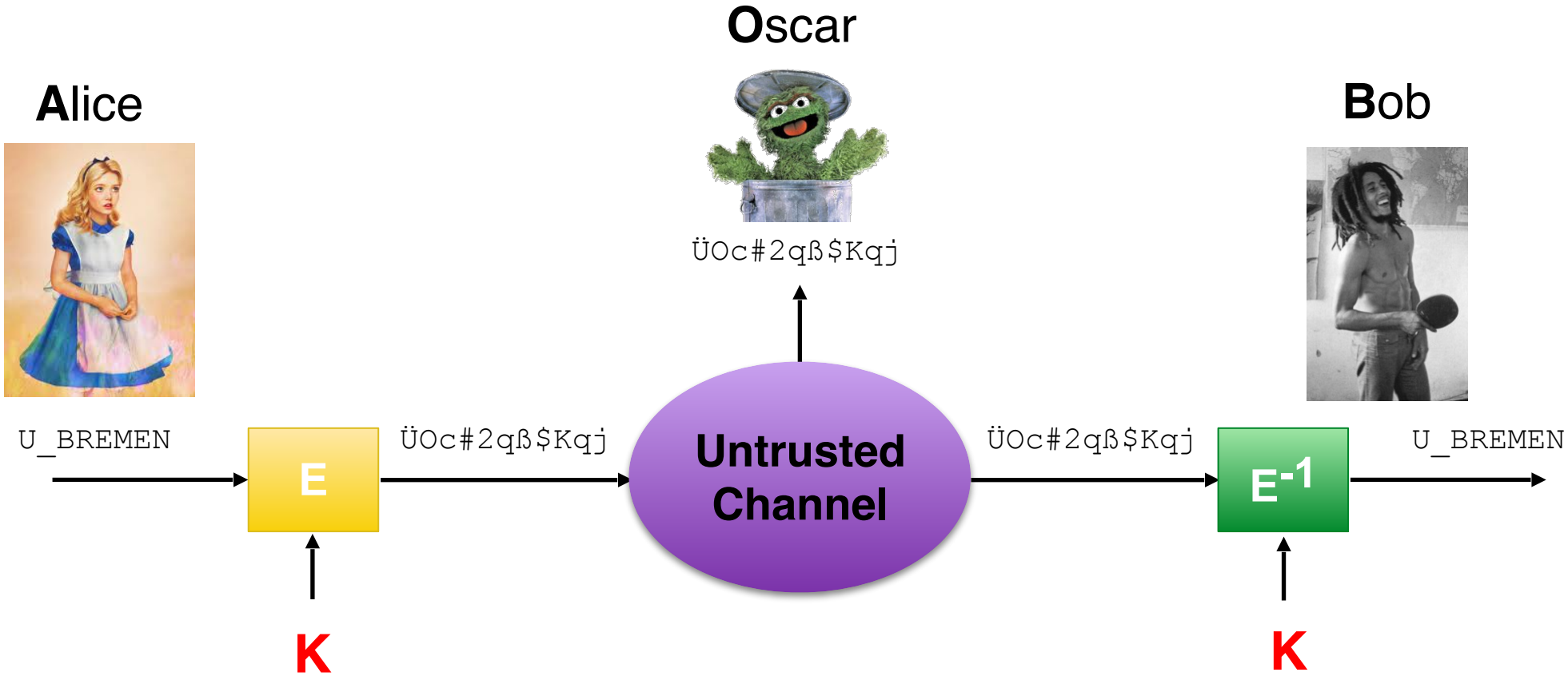
Mit Folien von Stefanie Gerdes und Tim Güneysu

Kryptologie = Kryptographie + Kryptoanalyse

- **Kryptographie**: Methoden zur Ver- und Entschlüsselung von Nachrichten und damit zusammenhängende Methoden
- **Kryptoanalyse**: Entschlüsselung ohne Zugriff auf den Schlüssel

- Kryptographie benötigt die Kryptoanalyse, um die erreichte Sicherheit der Verfahren beurteilen zu können

Symmetrische Verschlüsselung



Terminologie

- Klartext (plaintext): **P**
- Chiffretext/Geheimtext/Kryptotext (ciphertext): **C**
- Schlüssel (key): **K**

- Nachricht (message): M
  - Wenn unwesentlich ist, ob verschlüsselt wird oder nicht

Kryptoanalyse: Angriffsklassen (1)

- Ziel: **Schlüssel** finden oder jede Information, die beim Ent- (oder Ver-)schlüsseln von neuem Text hilft

- Chiffretext-Angriff (***ciphertext-only attack***):
  - Mehrere Chiffretexte stehen dem Angreifer zur Verfügung
  - Nutzt statistische Eigenschaften zwischen Chiffretexten aus; ggf. auch die des zugrunde liegenden Klartextes (sofern bekannt)

- Angriff mit bekanntem Klartext (***known-plaintext attack***):
  - Sowohl Chiffretext/Klartextpaare stehen zur Verfügung

Kryptoanalyse: Angriffsklassen (2)

- Angriff mit gewähltem Klartext (***chosen-plaintext attack***):
  - Erzeuge eine Reihe von Klartexten
  - Erhalte die dazugehörigen Chiffretexte

- Angriff mit adaptiv gewähltem Klartext (oder Chriffretext) (***adaptive chosen-plaintext (or ciphertext) attack***):
  - Führe verschiedene Angriffe mit gewähltem Klartext (Chiffretext) durch
  - Nutze das Wissen der zuvor durchgeführten Angriffe zur Erzeugung von neuen Klartexten

Historisch

Historische Kryptographie

(heute zu leicht zu brechen)

Historisch

Transposition

- Transposition:
  Verschieben von Buchstaben innerhalb des Textes
- Beispiel: Gartenzaun-Verschlüsselung

D E E D N T I F M R E M T A E N
I S N U G R F T O G N I T G I

- Problem: Verfahren muss geheim gehalten werden

Historisch

Transposition

- Skytale: Lederstreifen wurde um Holzstück *bestimmter Größe* gewickelt und beschriftet

Historisch

Caesar-Chiffre

- Caesar-Chiffre : Jeder Buchstabe wird durch drittnächsten ersetzt (tim→ wlp):
  ((x + 3) % 26)

P=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
C=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

  - Beispiel: PRUJHQ IUXHK
  - Variante der Caesar-Chiffre: ROT13 ((x + 13) % 26)

Kerckhoffs-Prinzip

- Kerckhoffs-Prinzip: Kryptosystem bleibt sicher, wenn alles (auch das Verfahren) außer einem **Schlüssel** bekannt

  Auguste Kerckhoffs, 1883

  - Schlüssel kann leicht/oft getauscht werden
  - Offenes kryptographisches Verfahren erlaubt externe Analyse

- Erweiterte Caesar-Chiffre: Addend als Schlüssel
  - Nur 25 (26–1) mögliche Schlüssel ⇒ **brute force noch zu einfach**

Historisch

Substitution

- Substitution: Ein Buchstabe wird durch einen anderen Buchstaben ausgetauscht
- Monoalphabetisch: Ein Buchstabe steht für genau einen anderen Buchstaben
- Problem wie bei der Caesar-Chiffre: Unwirksam, sobald Verfahren/Permutation bekannt ist

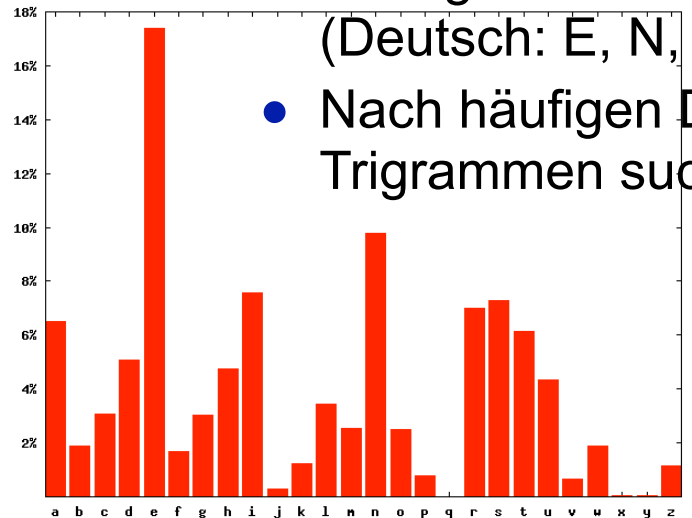
Historisch

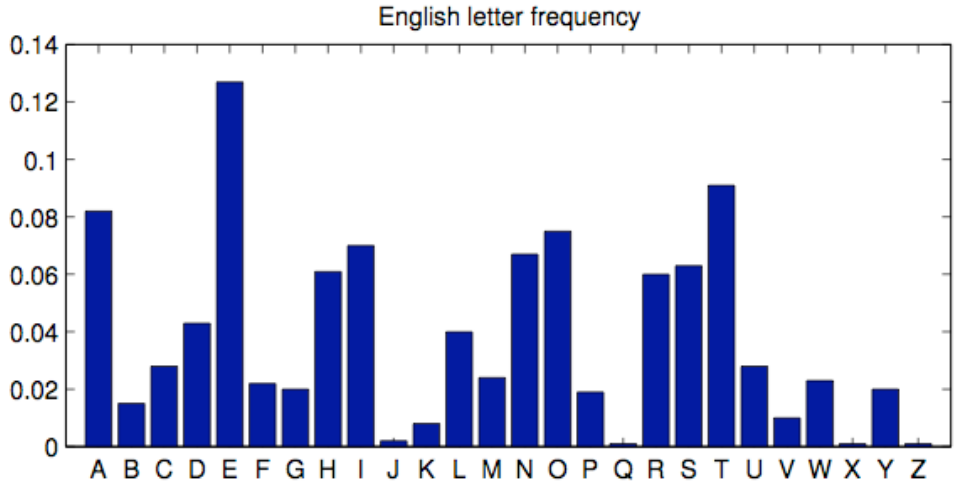
Historie: Substitutionschiffre

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

Monoalphabetische Substitutionschiffren

- Schlüssel ist Permutation K: A ↔ A
  - 26! > 4 × 10^{26} verschiedene Schlüssel
  - Brute force ist bereits schwierig
- Statistik-Angriff:
  - Häufigste Buchstaben suchen (Deutsch: E, N, I, S, R, A, …)
  - Nach häufigen Digrammen/ Trigrammen suchen





Historisch

Polyalphabetische Chiffren

P=	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
C1=	Y	X	B	C	U	R	P	H	N	O	M	E	K	J	T	Z	W	I	G	D	A	F	Q	L	S	V
C2=	Q	T	V	Y	R	X	A	J	W	N	U	C	I	F	L	M	B	E	H	Z	P	D	S	G	O	K

„Vigenère“-Chiffre:

- $K = k_0 k_1 k_2 \ldots k_{n-1}$
- $P = p_0 p_1 p_2 \ldots p_{m-1}$
- $c_i = (p_i + k_{(i \,\%\, n)}) \,\%\, 26$ (via Codewort)
- Statt Modulo-Addition kann XOR oder jeder andere Gruppen-Operator verwendet werden
- Analyse: *n* finden; Angriff auf Stellen mit gleichem *(i % n)*
- Verallgemeinerung: Änderung der Substitutionsregel bei jedem Zeichen

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

Ende des historischen Einschubs

Perfekte Chiffren

- Angreifer erhält keine Informationen aus dem Chiffretext
  - Kenntnis eines Chiffretextes ändert nichts an der statistischen Wahrscheinlichkeit möglicher Klartexte
- Einzige Lösung: One-time-pad
  - Schlüssel ist genauso lang wie Klartext und darf nur exakt einmal verwendet werden
  - Bietet informationstheoretische Sicherheit
  - $c_i = p_i \oplus k_i$ (XOR — jede andere Gruppe geht auch)
  - Problem: Wie Schlüssel transportieren?
    - Wiederverwendung ist extrem gefährlich

Blockchiffren: DES und AES

- Feste Blockgrößen und Schlüsselgrößen
  - DES: 64 bit pro Block, 56 bit pro Schlüssel
  - AES: 128 bit pro Block, 128, 192 oder 256 bit pro Schlüssel
- Produktchiffre (rundenbasierte Algorithmen)
  - DES: 16 Runden, AES (Rijndael): 10–14 Runden
- DES gilt als unsicher, 3DES = $E_{K1}(D_{K2}(E_{K3}(P)))$ (168 bit)
- AES wurde als Nachfolger von DES entwickelt
  - Offenes Auswahlverfahren
  - Viele Kryptoanalyse-Versuche vor und nach der Auswahl

➔ **Details in der „Einführung in die Kryptographie“**

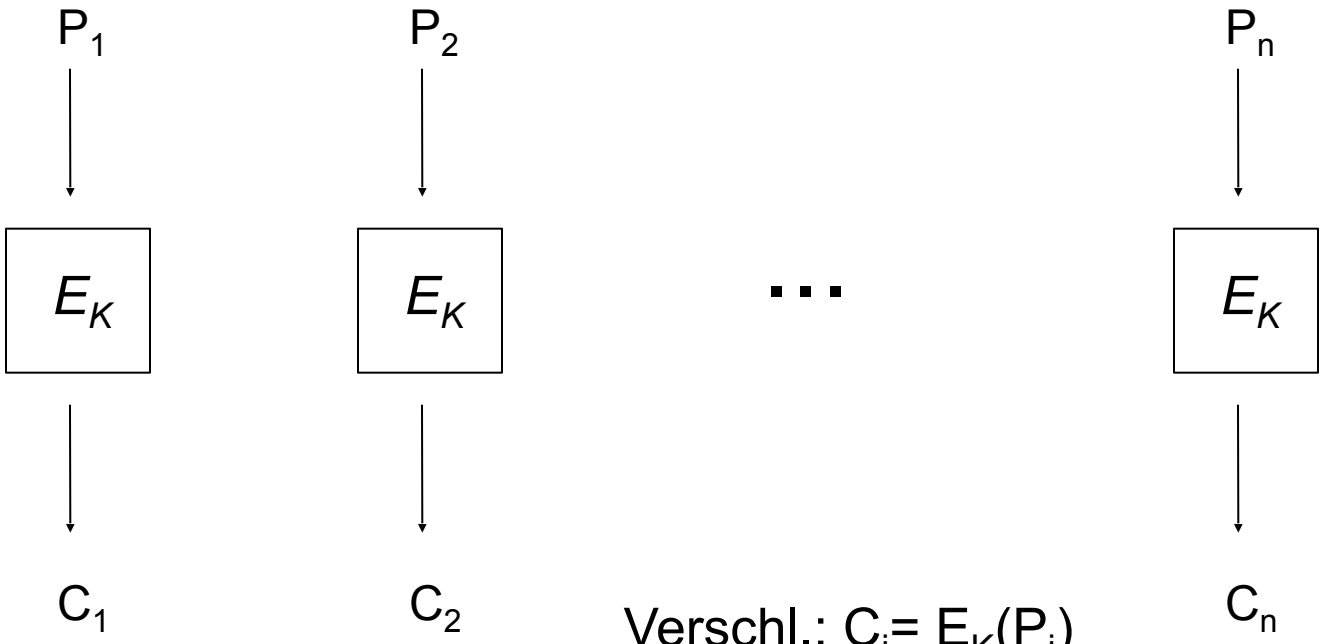
Nutzung von Blockchiffren: **Modes of operation** (Betriebsarten)

- **ECB** (Electronic Codebook): Je n Bit des Klartextes werden einzeln für sich verschlüsselt

- Problem: Identitäten/Redundanzen im Klartext werden im Chiffretext sichtbar
  - Kryptanalyst kann sich mit known-plaintext “Bausteine” schaffen
  - Wiederholte Nachrichten sind leicht zu erkennen

Funktionsweise: ECB

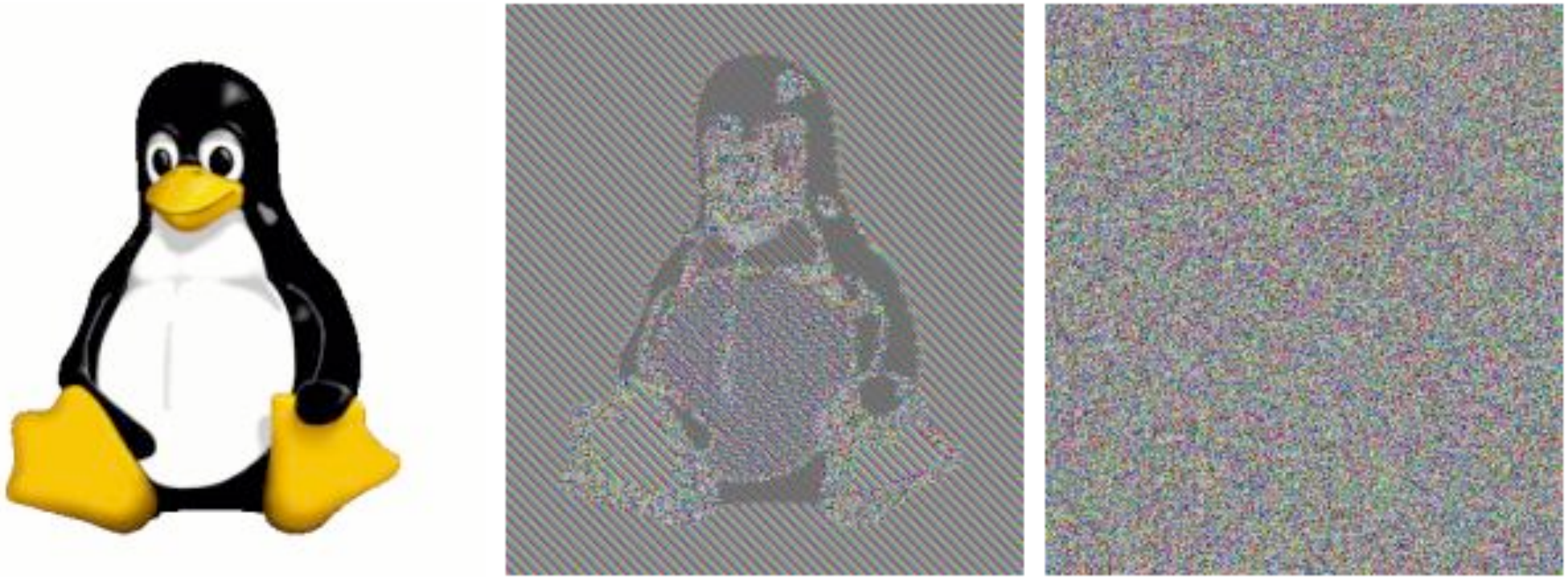
- Je n Bit des Klartextes (= Block) werden einzeln für sich verschlüsselt



Verschl.: $C_i = E_K(P_i)$

Entschl.: $P_i = D_K(C_i)$

ECB: Visualisierung



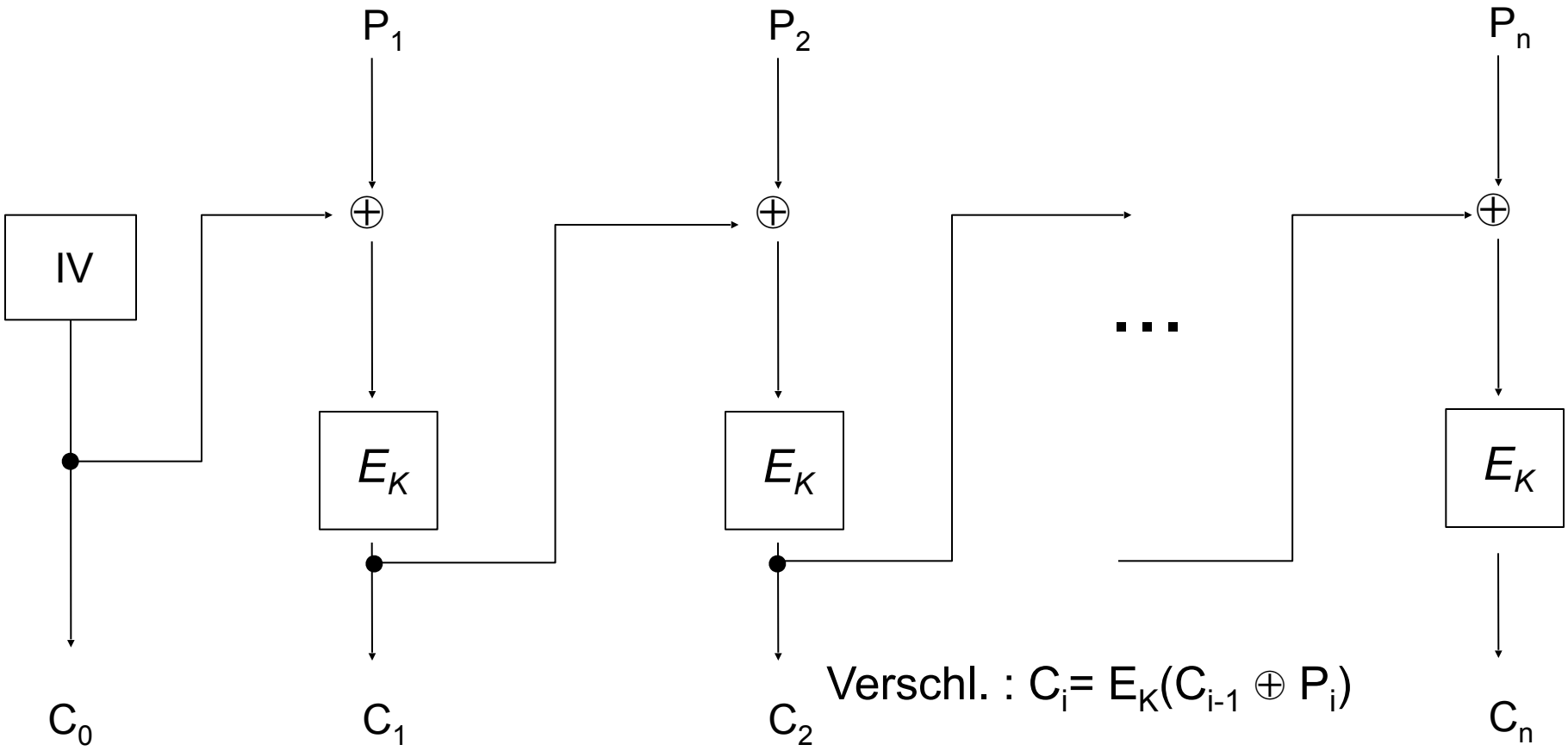
Original *Encrypted using ECB mode* *Encrypted securely*

[Larry Ewing lewing@isc.tamu.edu The GIMP]

CBC: Cipherblock Chaining

- Vor der Verschlüsselung wird P_i mit C_{i-1} verändert (XOR)
- Zum Schutz von P_1 wird zu Beginn eine Zufallszahl vorgegeben (Initialization Vector, IV)
- Bitfehler zerstört aktuellen Block und führt zu Bitfehler im Folgeblock

Funktionsweise: CBC



Verschl. : $C_i = E_K(C_{i-1} \oplus P_i)$

Entschl.: $C_i = C_{i-1} \oplus D_K(C_i)$

Padding

- Input ist nicht immer genau n Blöcke groß
- Auffüllen mit Padding
- Wie wieder entfernen?
- ➔ Immer Padding,
  - z.B.: letztes Byte sagt die Anzahl der Bytes (SSLv3)
  - Besser: PKCS #7: Alle Padding-Bytes mit dem selben Wert:

Beispiel mit 64 Bit Blockgröße

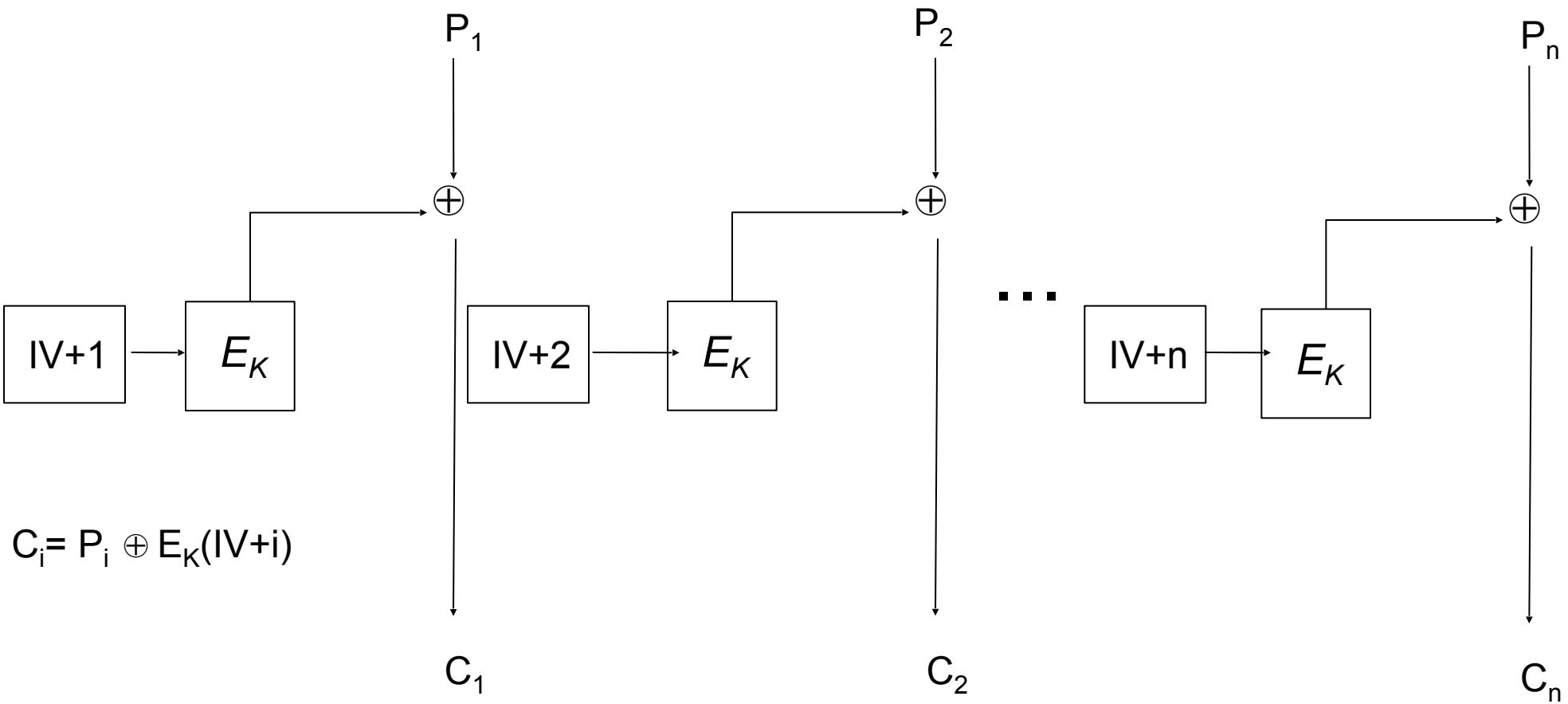
	BLOCK #1								BLOCK #2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Ex 1	F	I	G													
Ex 1 (Padded)	F	I	G	0x05	0x05	0x05	0x05	0x05								
Ex 2	B	A	N	A	N	A										
Ex 2 (Padded)	B	A	N	A	N	A	0x02	0x02								
Ex 3	A	V	O	C	A	D	O									
Ex 3 (Padded)	A	V	O	C	A	D	O	0x01								
Ex 4	P	L	A	N	T	A	I	N								
Ex 4 (Padded)	P	L	A	N	T	A	I	N	0x08	0x08	0x08	0x08	0x08	0x08	0x08	0x08
Ex 5	P	A	S	S	I	O	N	F	R	U	I	T				
Ex 5 (Padded)	P	A	S	S	I	O	N	F	R	U	I	T	0x04	0x04	0x04	0x04

Weitere Stromchiffren-Modi

- **CTR** (Counter Mode):
  $R_i = E_K(i + O)$, $C_i = P_i \oplus R_i$
- Offset O wird wie IV mit Chiffretext übertragen
- Vorteil: Direktzugriff in die Mitte des Stroms

- **CFB** (Cipher Feedback Mode):
  $R_i = E_K(C_{i-1})$, $C_i = P_i \oplus R_i$
  - Ein Bitfehler im Chiffretext erzeugt einen Bitfehler und zerstört den Folgeblock
- C_0 wird als IV mit Chiffretext übertragen

Funktionsweise: CTR



Angriffe gegen Stromchiffren

- Modifizierte Wiedereinspielung (vor allem OFB/CTR)
  - Ist Struktur des Klartextes bekannt, reicht evtl. ein „Bitfehler“
  - Schutz: Integritätsprüfung
- Bei Wiederverwendung der IV/K-Kombination:
  - $E(A) \oplus E(B) = (A \oplus R) \oplus (B \oplus R) = A \oplus B$
  - Schutz: K oft genug wechseln (und IV nie wiederholen)

Schlüsselerzeugung

Qualität von Zufallszahlen

https://dilbert.com/strip/2001-10-25

Entropie

- Entropie in der Informationstheorie (Shannon 1948):
  Informationsgehalt
  „Maß der Unsicherheit“ (Überraschung)
- N völlig zufällige Bits ➔ Entropie N bit
- Werden Bits vorhersagbar, sinkt die Entropie
  Redundanzen
  Statistische Regelmäßigkeit
- Verlustlose Kompression kann auf Entropie, aber nicht weiter komprimieren

Zufallszahlen

- Wichtig für Sitzungsschlüssel etc.
  - Viele Angriffe basieren auf der Möglichkeit, Zufallszahlen zu raten

- True RNG*): *Entropie* wächst mit jedem Bit
- Pseudo-RNG, Deterministischer RNG:
  Entropie steckt im “Seed” (Startwert), dann konstant
  - Hybridformen: kontinuierliches Seeding

*) RNG = Random Number Generator

Zufallszahlen

- Wie generieren?
- Physikalische Prozesse und Entropiequellen
  - Thermisches Rauschen, Jitter, Metastabile Prozesse, radioaktive Zerfallsprozesse
- Behelfsweise zu verwenden
  - Computerhardware (Festplatten, Sound- und Videokarten)
  - Benutzereingaben

Zufallszahlen

- Nachteile von Hardware für Zufallszahlen:
- Oft langsam
- Nicht immer verfügbar
- Ohne Software-Hilfe sehr aufwendig

Zufallszahlen

- Linux Entropiepool
- Benutzer: Tastatur, Maus
- Hardware: Netzlast, Interrupts

Linux: /dev/random

Aber: immer noch sehr langsam!

Zufallszahlen

- /dev/urandom:
- Nimmt Daten aus Entropie-Pool
- Blockiert nicht auf leerem Pool
- Erzeugt dann selbst Zufallszahlen via DRNG*)

*) DRNG: Deterministic Random Number Generator
~ DRBG: Deterministic Random Bit Generator

Zufallszahlen

- Pseudozufallszahlengenerator
  (Pseudo Random Number Generator, PRNG)
- Erzeugt aus einem Initialisierungswert (seed) immer gleiche Folge von Zufallszahlen

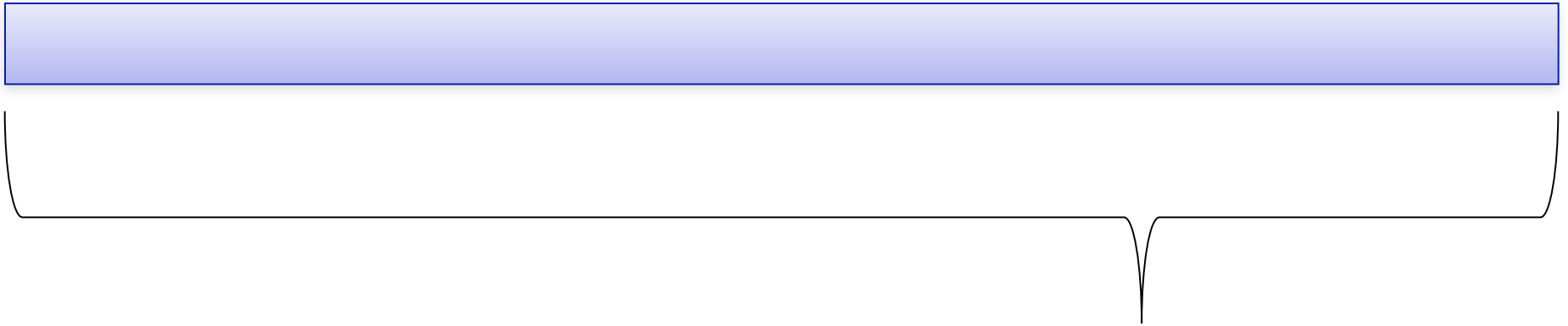
Block-Chiffre als Pseudo-Zufallszahlengenerator

- Kryptographisch starker PRNG: $R_i = E_K(R_{i-1})$
- **OFM** (Output Feedback Mode):
  $C_i = P_i \oplus R_i$ (mit $R_i = E_K(R_{i-1})$)
- **Achtung**: Wiederverwendung von K/R_0 = 

Hash-Funktionen

Kryptographischer Hash

- Hash dampft variabel lange Binärdaten in Fingerabdruck ein:



- Beispiele: MD5, SHA-1, SHA-256:
  - ~~MD5: 128 bit~~
  - SHA-1: 160 bit (obsolet)
  - SHA-256: 256 bit

Beispiele (ASCII-Strings → Hex)

- sha1("") ➔ da39a3ee5e6b4b0d3255bfef95601890afd80709
- sha1("a") ➔ 86f7e437faa5a7fce15d1ddcb9eaeaea377667b8
- sha1("b") ➔ e9d71f5ee7c92d6dc9e92ffdad17b8bd49418f98
- sha1("ab") ➔ da23614e02469a0d7c7bd1bdab5c9c474b1904dc
- sha1("Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Hi omnes lingua, institutis, legibus inter se differunt. Gallos ab Aquitanis Garumna flumen, a Belgis Matrona et Sequana dividit.") ➔ c92db729e51319e19d26831ac4896b55f955ccb9

Was ist ein guter kryptographischer Hash?

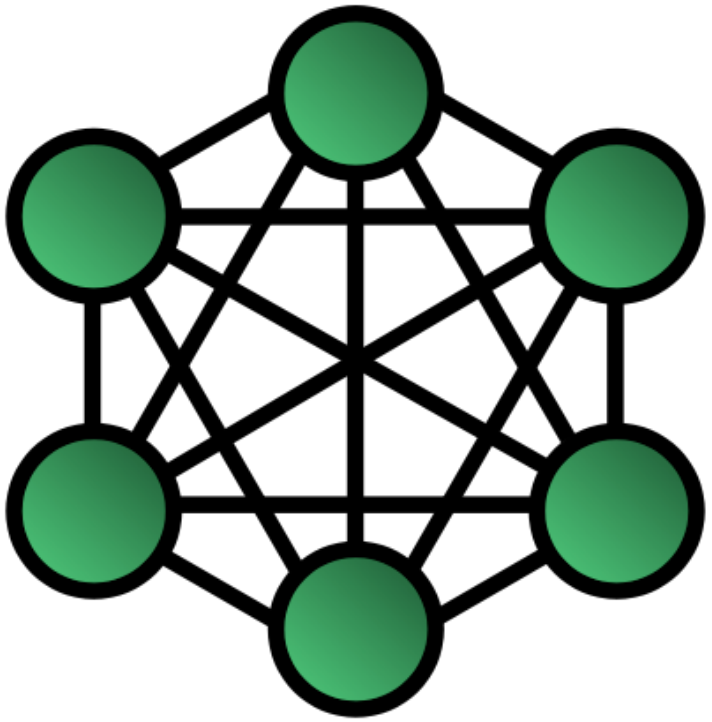
- Schwierig zu erreichendes Ziel:
  - Kollisionsresistent: schwer, $x \neq y$ zu finden mit $h(x) = h(y)$
    - ✳Zwei Personen finden, die am gleichen Tag Geburtstag haben
- Etwas weniger problematisch in der Sicherheitsbetrachtung:
  - Urbildresistent: schwer, zu einem a ein y zu finden mit $a = h(y)$
    - ✳ Zu einem gegebenen Geburtsdatum die passende Person finden
  - Urbild-2: schwer, zu einem x ein $y \neq x$ zu finden mit $h(x) = h(y)$
    - ✳ Eine zweite Person finden, die am gleichen Tag Geburtstag hat

Geburtstagsparadoxon (0)

- 23 Personen sitzen in einem Raum
- Wie hoch ist die Wahrscheinlichkeit, dass eine dieser Personen am selben Tag wie ich Geburtstag hat?
- Wie hoch ist die Wahrscheinlichkeit, dass zwei von diesen Personen am selben Tag Geburtstag haben?

Geburtstagsparadoxon (1)

- Mit 6,1% Wahrscheinlichkeit hat eine Person von 23 am selben Tag Geburtstag
- Von 23 Personen haben mit 50 % Wahrscheinlichkeit zwei am selben Tag Geburtstag
- n x (n – 1)/2 Verbindungen
- Allgemein: aus k möglichen Werten reichen $k^{1/2}$ für ca. 50 % Kollisionswahrscheinlichkeit



Einfluss des Geburtstagsparadoxons

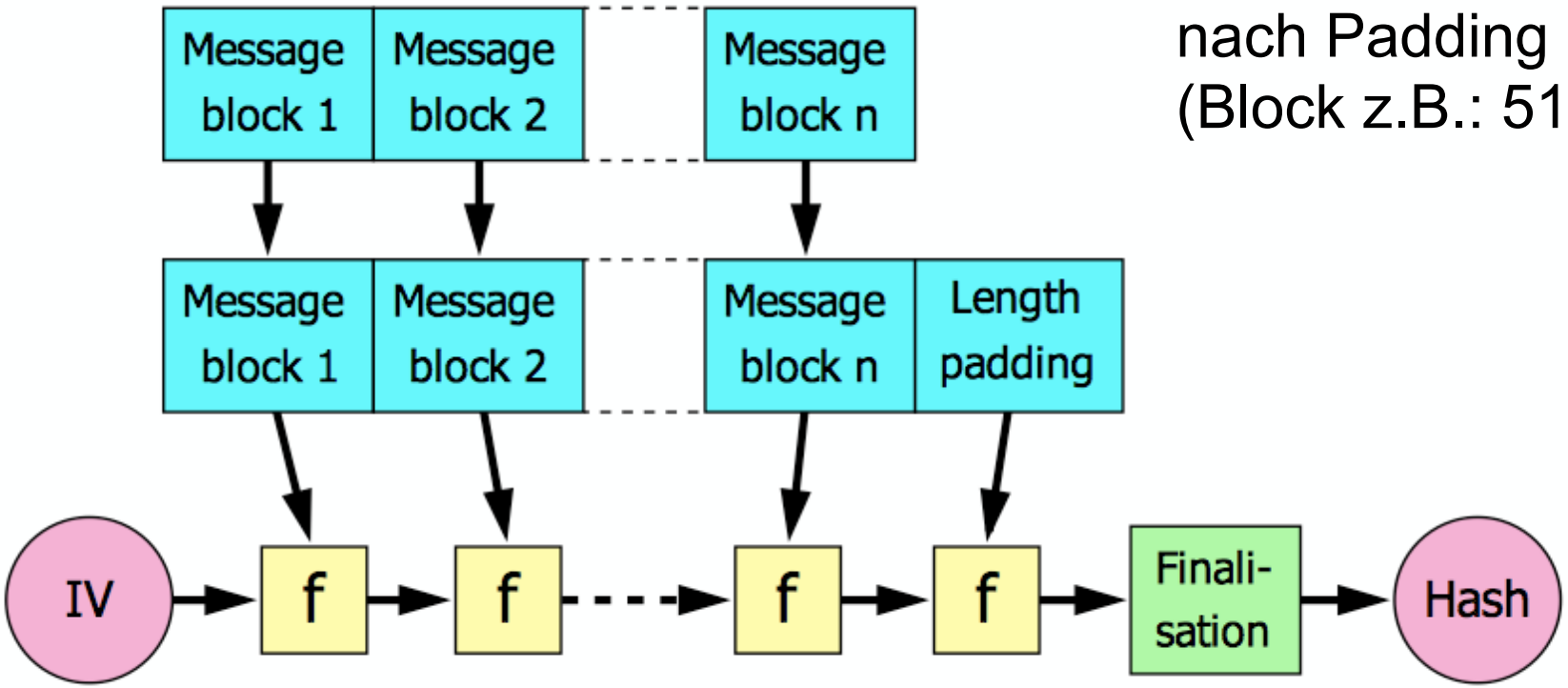
- Erinnerung: Von 23 Personen haben mit 50 % Wahrscheinlichkeit zwei am selben Tag Geburtstag
  - Allgemein: aus k möglichen Werten reichen $k^{1/2}$ für ca. 50 % Kollisionswahrscheinlichkeit
- SHA-1: 2^{160} Möglichkeiten
  - 2^{80} Versuche ergeben mit $p \geq 0.5$ Kollision („Geburtstagsangriff“)

https://shattered.it/static/shattered.pdf

- Neue Angriffe auf SHA-1 reduzieren 2^{80} auf 2^{63}
  - Immer noch ähnlich 2^{64} (MD5 vor den Angriffen)
  - Übergang auf SHA-2 (SHA-256, SHA-512, …)

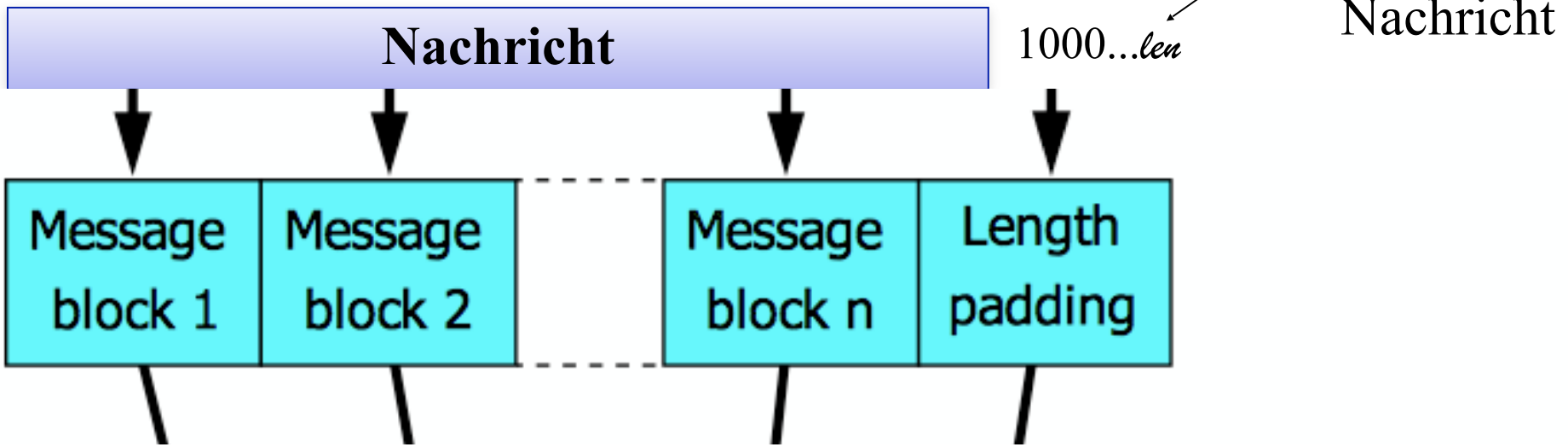
Wir bauen uns einen Hash: Merkle-Damgård

$H_i = f(H_{i-1}, M_i)$,
nach Padding
(Block z.B.: 512 bit)



Padding: Auffüllen auf Blockgröße

- (Am Beispiel MD5 oder SHA-1:)



Status von Hash-Funktionen

Merkle-Damgård

Hash-Funktion	Status, Haltbarkeitsdauer
MD5	Nicht mehr Kollisionsresistent, nur noch aus Rückwärtskompatibilität benutzen
RIPEMD-160	Europäische Spezialität, (~ SHA-1)
SHA-1	„Shattered“, war nur noch bis 2010 „sicher“
SHA-2 (SHA-256, SHA-512)	Z.Z. Funktionen der Wahl
„SHA-3“	Auswahlprozess (vgl. AES) ➔ **Keccak**

Sponge

SHA-3 Competition

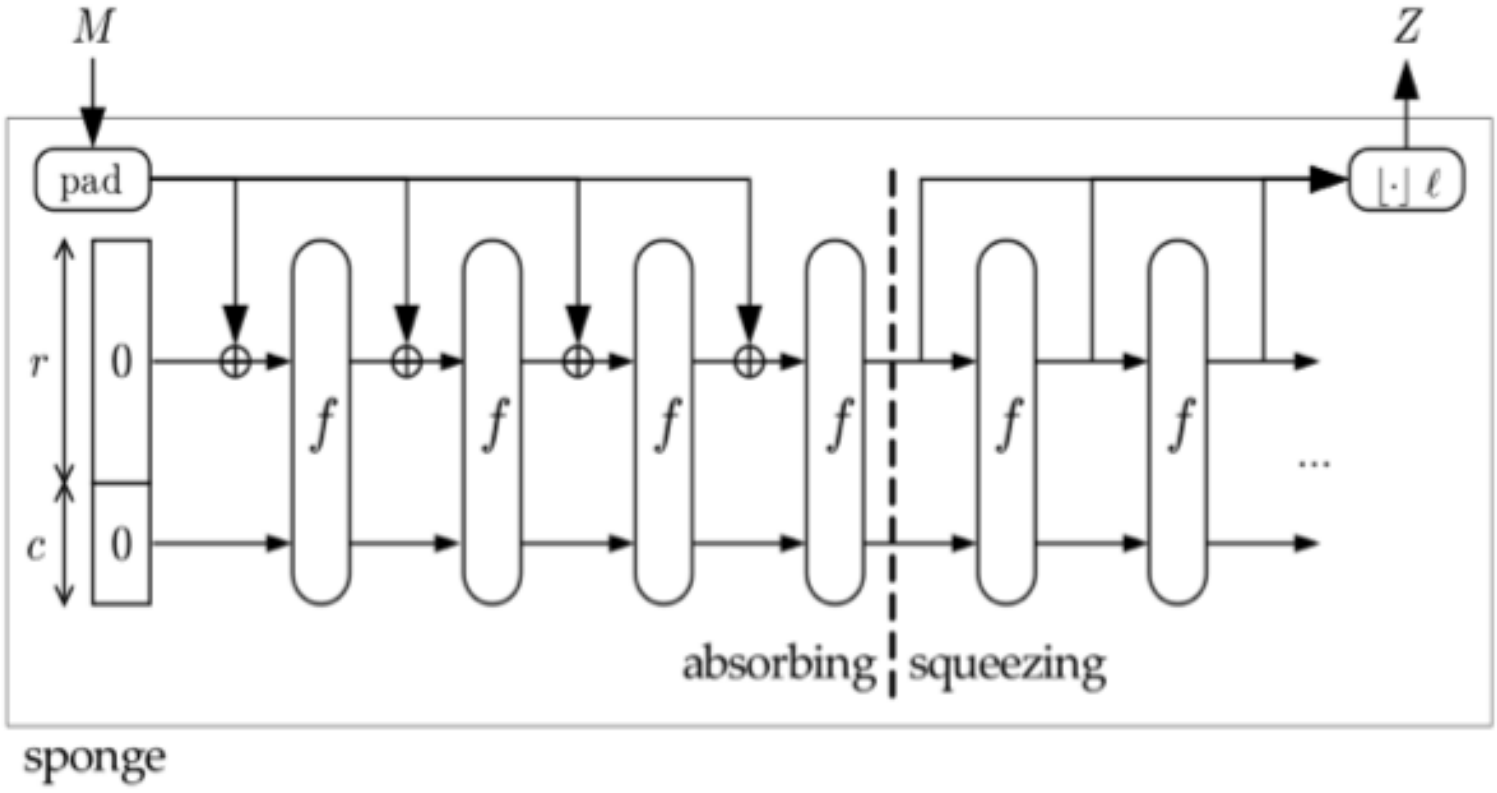
11/2/2007	SHA-3 Competition Began.
10/2/2012	*Keccak announced as the SHA-3 winner.*

Secure Hash Algorithms Outlook

- SHA-2 looks strong.
- We expect Keccak (SHA-3) to co-exist with SHA-2.
- Keccak *complements* SHA-2 in many ways. Keccak is good in different environments.

Keccak is a sponge - a different design concept from SHA-2.

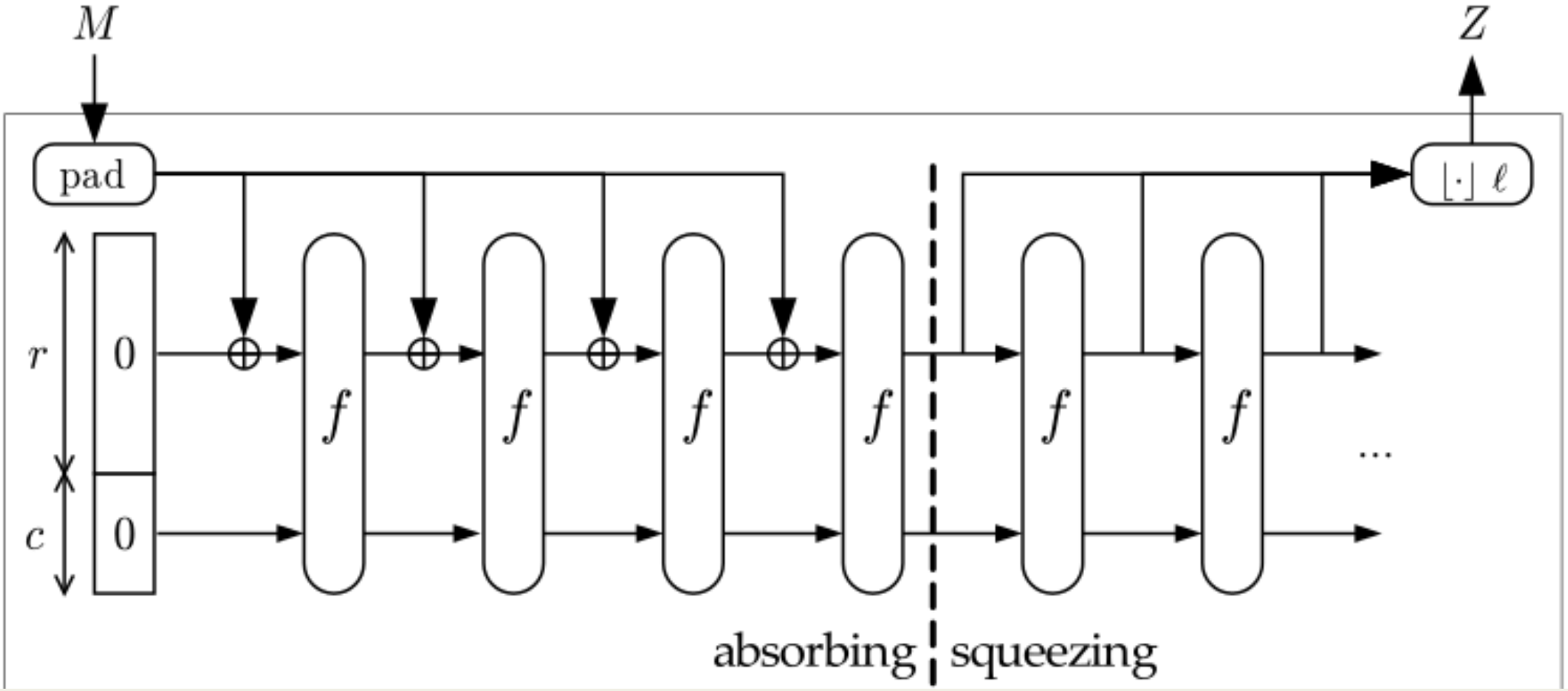
Sponge Construction



sponge

Sponge capacity corresponds to a security level: s = c/2.

Sponge mode:
b = r (bitrate) + c (capacity)

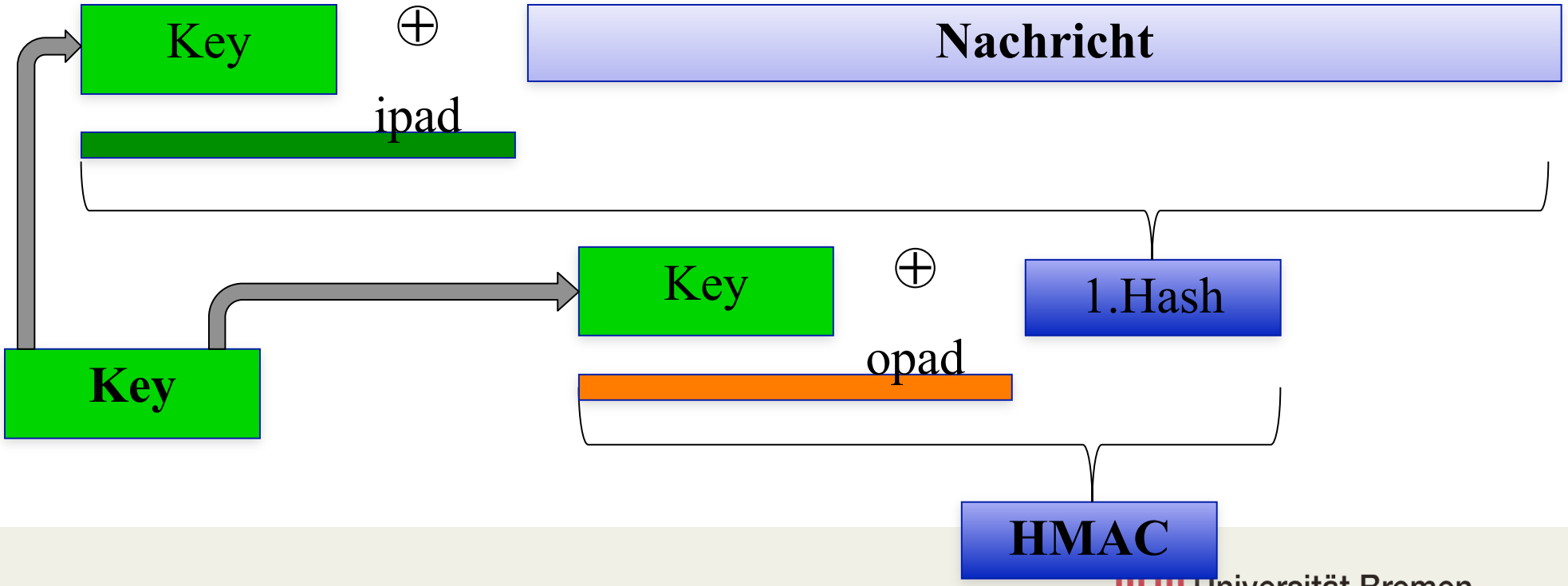


Message Authentication Codes (MAC)

- Nicht jede Nachricht möchte man signieren (aufwendig)
- Idee:
  - Erst gemeinsames Geheimnis vereinbaren
  - Sender verbindet dieses dann mit Nachricht kryptographisch
  - Empfänger wendet die gleiche Funktion an
  - Nachricht muß echt sein, wenn Ergebnis übereinstimmt
- „Hash mit Schlüssel“ / Blockchiffren zur Nachrichtenauthentisierung

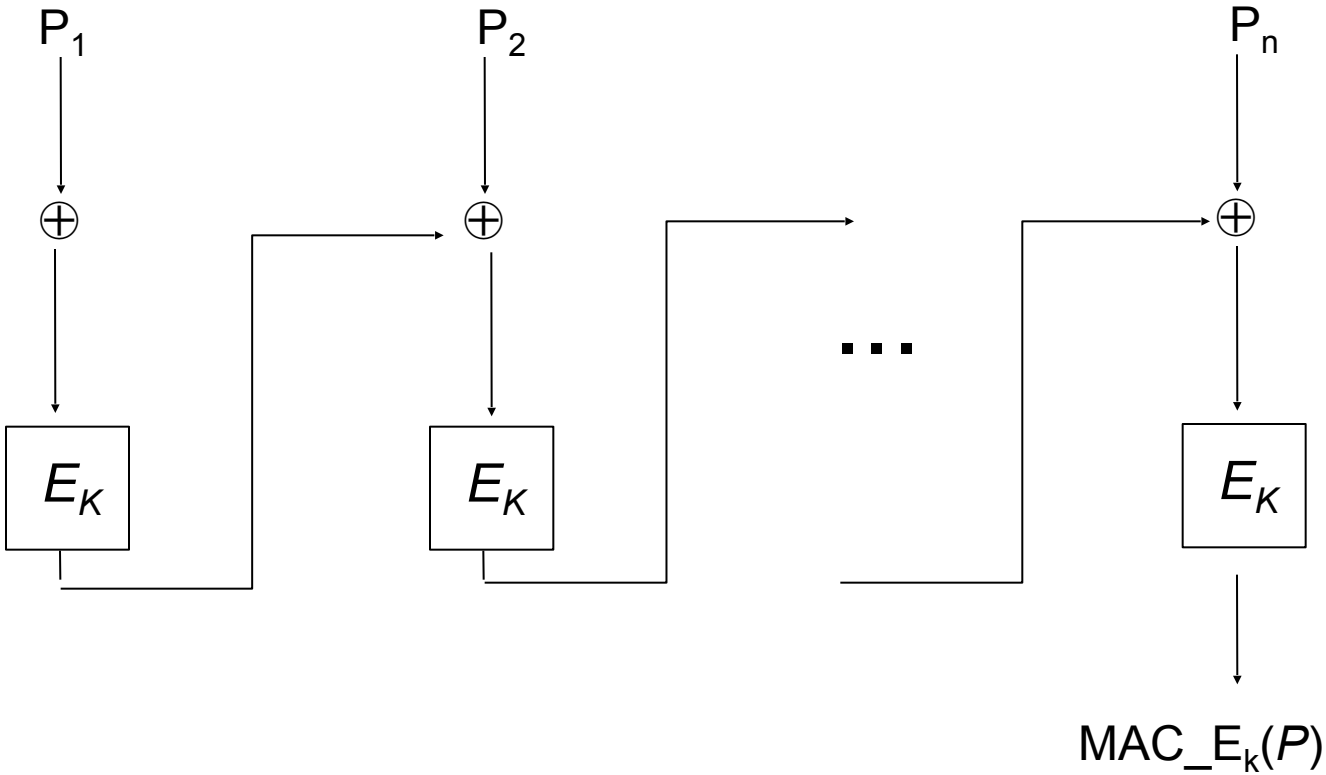
Einsatz von Hash-Funktionen für MACs: **HMAC**

- $HMAC_K(M) = h(K \oplus opad \,||\, h(K \oplus ipad \,||\, M))$ (RFC 2104)



Exkurs: CBC-MAC (Message Authentication Code)

- Integrität:
  Hash über Nachricht
- Kann nur mit Schlüssel K erstellt oder verifiziert werden



Kombination MAC/ Verschlüsselung

- Stromchiffre braucht immer Integritätsprüfung
- Kombination möglich?
- OCB: Offset Codebook
  - Literatur:
    - Rogaway, Bellare, Black, Krovetz:
      OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption
- Problem: Patentiert
- 802.11i verwendet stattdessen CTR + CBC-MAC
  - Details: RFC 3610

Authenticated Encryption with Associated Data (AEAD)

Authenticated Encryption

- Vertraulichkeit + Integrität + Authentizität
- effizient mit Block-Verschlüsselung realisierbar
  → benötigt Counter und CBC-MAC

Mechanismus: RFC 5116

- Verschlüsselung: $K \times N \times P \times A \rightarrow A \times C$
- Entschlüsselung: $K \times N \times A \times C \rightarrow A \times P$
- AEAD-Algorithmus legt Länge von K, K_LEN, fest

K	Key	A	Associated Data	P	Plaintext
N	Nonce	C	Ciphertext		

Wie sicher ist mein Kryptosystem?

- Aussage: man müsste zum Brechen den gleichen Aufwand treiben wie für einen Brute-Force-Angriff mit 2^n Schritten
- n ist die Sicherheitsstufe in “Bits”
- z.B. AES-128: ungebrochen, also bräuchte man 2^{128} Schritte für eine Brute-Force-Angriff
  - “128-Bit-Sicherheit”
- Bei anderen Verfahren ist der Zusammenhang nicht immer so einfach!

Levels of Security

Table 7.1: Minimum symmetric key-size in bits for various attackers.

Attacker	Budget	Hardware	Min security
"Hacker"	0	PC	53
	< $400	PC(s)/FPGA	58
	0	"Malware"	62
Small organization	$10k	PC(s)/FPGA	64
Medium organization	$300k	FPGA/ASIC	68
Large organization	$10M	FPGA/ASIC	78
Intelligence agency	$300M	ASIC	84

[ECRYPT II 2011]

Levels of Security

Level (Bits)	Encryption	Hash	RSA, D-H	ECC
80	(3DES)	(SHA-1)	1024	160
112	(3DES)	SHA-224*), SHA-256*)	2048	224
128	AES-128	SHA-256*)	3072	256
192	AES-192	SHA-384*)	7680	384
256	AES-256	SHA-512	15360	521

*) oder SHA-512/nnn