

# IT Sicherheit - Vertiefung

Dozent: Prof. Dr. Michael Eichberg  
Kontakt: michael.eichberg@dhbw.de  
Version: 1.0.2

---

Folien: <https://delors.github.io/sec-schulung-fortgeschrittene-ein-tag/folien.de.rst.html>  
<https://delors.github.io/sec-schulung-fortgeschrittene-ein-tag/folien.de.rst.html.pdf>  
Fehler melden: <https://github.com/Delors/delors.github.io/issues>

# 1. Rückblick

# Themen der Grundlagenschulung

- Cybersecurity
- Schutzziele der IT Sicherheit
- Social Engineering Angriffe
- NIS 2 (ganz kurz)
- CVSS/CVE/KVE
- Schwachstellenmanagement
- ein grober, aller erster Überblick über Verschlüsselungsverfahren - Brute-Force - Hashing
- Passwortsicherheit

## 2. Ausblick

# Themenblöcke

- Verschlüsselungsalgorithmen
- Public-Key Verschlüsselung
- Netzwerksicherheit

# 3. Verschlüsselungsalgorithmen



# Definitionen

**Klartext:**

 *Plaintext*

Die Originalnachricht, die verschlüsselt werden soll.

**Geheimtext oder Chiffretext oder Kryptogramm:**

 *Ciphertext*


Die kodierte/verschlüsselte Nachricht.

**Verschlüsselung:**

 *Encryption*

Der Prozess der Umwandlung von Klartext in Geheimtext.

**Entschlüsselung:**


 *Decryption*

Der Prozess der Wiederherstellung des Klartextes aus dem Geheimtext.



# Definitionen

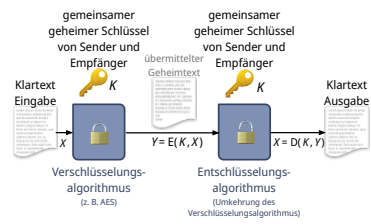
**Kryptographie:**  *Cryptography*  
Das Studiengebiet der Verschlüsselungsschemata.

**Kryptoanalyse:**  *Cryptanalysis*  
Methoden und Techniken, die zur Gewinnung von Informationen aus einer verschlüsselten Nachricht dienen.

Analyse von kryptographischen Verfahren.

**Kryptologie:**  *Cryptology*  
Die Bereiche Kryptographie und Kryptoanalyse.

# Vereinfachtes Modell der symmetrischen Verschlüsselung

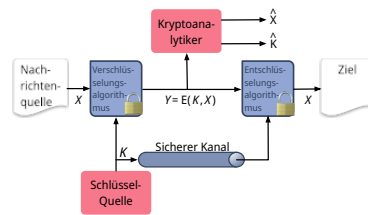


# Symmetrisches Verschlüsselungsmodell

Es gibt zwei Voraussetzungen für die sichere Verwendung der herkömmlichen Verschlüsselung:

1. Ein starker Verschlüsselungsalgorithmus.
2. Effektive Schlüsselverwaltung:
  - a. Sender und Empfänger müssen Kopien des geheimen Schlüssels auf sichere Weise erhalten haben und
  - b. den Schlüssel sicher aufbewahren.

# Modell eines symmetrischen Kryptosystems




# Kryptografische Systeme können entlang dreier unabhängiger Dimensionen charakterisiert werden

1. Die Art der Operationen, die zur Umwandlung von Klartext in Chiffretext verwendet werden.

- Substitution
- Transposition (Vertauschungen)

## Bemerkung

Eine Permutation ist eine Folge von Vertauschungen ( *Transposition*).

2. Die Anzahl der verwendeten Schlüssel.

Symmetrisch: Ein-Schlüssel-, **Secret-Key**-, konventionelle Verschlüsselung

Asymmetrisch: Zwei-Schlüssel- oder **Public-Key**-Verschlüsselung

3. Die Art und Weise, in der der Klartext verarbeitet wird:

- Blockchiffre
- Stromchiffre

# Kryptoanalyse und Brute-Force-Angriff

## Kryptoanalyse

- Der Angriff beruht auf der Art des Algorithmus und einer gewissen Kenntnis der allgemeinen Merkmale des Klartextes.
- Der Angriff nutzt die Eigenschaften des Algorithmus aus, um zu versuchen, einen bestimmten Klartext zu entschlüsseln oder den verwendeten Schlüssel zu ermitteln.

## Brute-force Angriff ( *brachiale Gewalt*)

- Der Angreifer probiert jeden möglichen Schlüssel an einem Stück Chiffretext aus, bis er eine verständliche Übersetzung in Klartext erhält.
- Im Durchschnitt muss die Hälfte aller möglichen Schlüssel ausprobiert werden, um Erfolg zu haben.

# Klassifizierung von Angriffen

Art des Angriffs	dem Kryptoanalytiker bekannt
Ciphertext Only	<ul style="list-style-type: none"><li>■ Verschlüsselungsalgorithmus und Chiffretext</li></ul>
Known Plaintext	<ul style="list-style-type: none"><li>■ Verschlüsselungsalgorithmus und Chiffretext</li><li>■ ein oder mehrere Klartext-Chiffretext-Paare, die mit dem geheimen Schlüssel verschlüsselt wurden</li></ul>
Chosen Plaintext	<ul style="list-style-type: none"><li>■ Verschlüsselungsalgorithmus und Chiffretext</li><li>■ Klartextnachricht, die vom Kryptoanalytiker gewählt wurde, zusammen mit dem zugehörigen Chiffretext, der mit dem geheimen Schlüssel verschlüsselt wurde.</li></ul>
Chosen Ciphertext	<ul style="list-style-type: none"><li>■ Verschlüsselungsalgorithmus und Chiffretext</li><li>■ Chiffretext, der vom Kryptoanalytiker gewählt wurde, zusammen mit dem zugehörigen entschlüsselten Klartext, der mit dem geheimen Schlüssel entschlüsselt wurde.</li></ul>
Chosen Text	<ul style="list-style-type: none"><li>■ Verschlüsselungsalgorithmus und Chiffretext</li><li>■ vom Kryptoanalytiker gewählte Klartextnachricht, zusammen mit dem zugehörigen Chiffretext, der mit dem geheimen Schlüssel verschlüsselt wurde.</li><li>■ vom Kryptoanalytiker gewählter Chiffretext zusammen mit dem entsprechenden entschlüsselten Klartext, der mit dem geheimen Schlüssel erzeugt wurde.</li></ul>

---

Das Ziel ist es immer den Schlüssel zu ermitteln, damit man weitere Kommunikation effektiv entschlüsseln kann.

# Sicherheit von Verschlüsselungsschemata

## *Bedingungslos Sicher* ( *Unconditionally Secure* )

- Unabhängig davon wie viel Zeit ein Gegner hat, ist es ihm unmöglich, den Geheimtext zu entschlüsseln, weil die erforderlichen Informationen nicht vorhanden sind.

## *Rechnerisch Sicher* ( *Computationally Secure* )

- Die Kosten für das Brechen der Chiffre übersteigen den Wert der verschlüsselten Informationen.
- Die zum Knacken der Chiffre benötigte Zeit übersteigt die Lebensdauer der Informationen.

### ? Frage

Wie lange könnte der Nutzen einer bestimmten Information andauern?



# Brute-Force Angriff

- Es werden alle möglichen Schlüssel ausprobiert, bis eine verständliche Übersetzung des Chiffriertextes in Klartext erreicht wird.
- Im Durchschnitt muss die Hälfte aller möglichen Schlüssel ausprobiert werden, um Erfolg zu haben.
- Zur Durchführung des Brute-Force-Ansatzes ist ein gewisses Maß an Wissen über den zu erwartenden Klartext erforderlich. Es werden Mittel zur automatischen Unterscheidung von Klartext und „Müll“ benötigt.

## ? Frage

Was bedeutet somit *bis eine verständliche Übersetzung des Chiffriertextes in Klartext erreicht wird*? Wenn der Klartext zum Beispiel ein Bild, ein Video oder ein Computerprogramm ist?



# Substitutionsverfahren

- Bei der Substitution werden die Buchstaben des Klartextes durch andere Buchstaben oder durch Zahlen oder Symbole ersetzt.
- Wenn der Klartext als eine Folge von Bits betrachtet wird, beinhaltet die Substitution das Ersetzen von Bitmustern des Klartextes durch Bitmuster des Geheimtextes.

# Cäsar Chiffre

- Einfachste und früheste bekannte Verwendung einer Substitutions-Chiffre; verwendet von Julius Cäsar.
- Dabei wird jeder Buchstabe des Alphabets durch einen Buchstaben ersetzt, der drei Stellen weiter hinten im Alphabet steht.
- Am Ende des Alphabets wird wieder am Anfang begonnen. Somit folgt auf den Buchstabe Z der Buchstabe A.

unverschlüsselt:	meet me after the toga party
verschlüsselt:	PHHW PH DIWHU WKH WRJD SDUWB

# Cäsar-Chiffre - historische Verwendung

Die Transformation kann wie folgt ausgedrückt werden:

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
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

Mathematisch, wenn wir jedem Buchstaben einen Wert zuweisen:

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Der Algorithmus zur Verschlüsselung ist dann ( $p$  ist der Wert des zu verschlüsselnden Buchstabens):

$$Y = E(3, p) = (p + 3) \bmod 26$$

# Verallgemeinerter Cäsar-Chiffre-Algorithmus

Eine Verschiebung kann beliebig groß sein ( $k$ ), so dass der allgemeine Caesar-Algorithmus lautet:

$$Y = E(k, p) = (p + k) \bmod 26$$

Wobei  $k$  einen Wert im Bereich von 1 bis 25 annimmt; der Entschlüsselungsalgorithmus ist einfach:

$$p = D(k, C) = (Y - k) \bmod 26$$

# Brute-Force-Kryptoanalyse der Caesar-Chiffre

Key	PHHW	PH	DIWHU	WKH	WRJD	SDUWB
1	OGGV	OG	CHVGT	VJG	VQIC	RCTVA
2	NFFU	NF	BGUFS	UIF	UPHB	QBSUZ
3	MEET	ME	AFTER	THE	TOGA	PARTY
4	LDDS	LD	ZESDQ	SGD	SNFZ	OZQSX
5	KCCR	KC	YDRCP	RFC	RMEY	NYPRW
6	JBBQ	JB	XCQBO	QEB	QLDX	MXOQV
7	IAAP	IA	WBPAN	PDA	PKCW	LWNPU
8	HZZO	HZ	VAOZM	OCZ	OJBV	KVMOT
9	GYYN	GY	UZNYL	NBY	NIAU	JULNS
10	FXXM	FX	TYMXK	MAX	MHZT	ITKMR
11	EWVL	EW	SXLWJ	LZW	LGYS	HSJLQ
12	DVVK	DV	RWKVI	KYV	KFXR	GRIKP
13	CUUJ	CU	QVJUH	JXU	JEWQ	FQHJO
...	...	...	...	...	...	...
25	QIIX	QI	EJXIV	XLI	XSKE	TEVXC

# Brute-Force-Kryptoanalyse (z. B. der Caesar-Chiffre)

Die Entschlüsselung ist komplizierter, wenn der Klartext bereits eine sehr hohe Entropie aufweist, wie z. B. im Falle einer komprimierten ZIP Datei:

00000000:	504b	0304	1400	0000	0800	afb1	4257	1da9	PK.....BW..
00000010:	b0b9	4b00	0000	4f04	0000	0800	1c00	6465	..K...O.....de
00000020:	6d6f	2e74	7874	5554	0900	036a	241b	65a4	mo.txtUT...j\$.e.
00000030:	a9c0	6575	780b	0001	04f8	0100	0004	1400	..eux.....
00000040:	0000	edcc	db09	8030	0c05	d07f	a7c8	049d	.....0.....
00000050:	a28b	c4f6	6203	e983	18d0	6e2f	ee91	ffc3	....b.....n/....
00000060:	c928	b697	cb1c	2437	f569	a032	fb52	29ec	.(....\$7.i.2.R).
00000070:	a8f4	340c	f206	5aca	321c	afff	8cd5	c075	..4...Z.2.....u
00000080:	d3c5	762a	d291	2389	2492	48d2	0750	4b01	..v*...#.\$..H..PK.
00000090:	021e	0314	0000	0008	00af	b142	571d	a9b0	.....BW...
000000a0:	b94b	0000	004f	0400	0008	0018	0000	0000	..K...O.....
000000b0:	0001	0000	00ff	8100	0000	0064	656d	6f2e	.....demo.
000000c0:	7478	7455	5405	0003	6a24	1b65	7578	0b00	txtUT...j\$.eux..
000000d0:	0104	f801	0000	0414	0000	0050	4b05	0600	.....PK...
000000e0:	0000	0001	0001	004e	0000	008d	0000	0000	.....N.....
000000f0:	00								

?

Frage

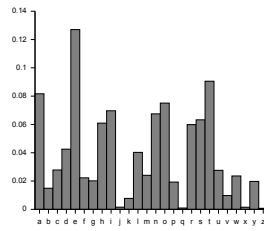
Wie kann man - wenn man weiss, dass es sich um eine ZIP Datei handelt - die Caesar-Chiffre knacken?



# Monoalphabetische Chiffren

- Eine Permutation einer endlichen Menge von Elementen  $S$  ist eine geordnete Folge aller Elemente von  $S$ , wobei jedes Element genau einmal vorkommt.
- Wenn die „Chiffre“-Zeile (siehe Cäsar-Chiffre) eine beliebige Permutation der 26 alphabetischen Zeichen sein kann, dann gibt es  $26!$  oder mehr als  $4 \times 10^{26}$  mögliche Schlüssel.
- Dies ist um 10 Größenordnungen größer als der Schlüsselraum für DES!
- Der Ansatz wird als monoalphabetische Substitutions-Chiffre bezeichnet, da pro Nachricht ein einziges Chiffre-Alphabet verwendet wird.

# Häufigkeit der englischen Buchstaben [1]



[1] Analyse des Concise Oxford Dictionary (9th edition, 1995) — <https://www.nd.edu>

# Angriffe auf Monoalphabetische Chiffren

Sie sind leicht zu knacken, da sie die Häufigkeitsdaten des ursprünglichen Alphabets widerspiegeln.

Die Gegenmaßnahme besteht darin, mehrere Substitute (Homophone) für einen einzigen Buchstaben anzubieten.

# Polyalphabetische Chiffren

Polyalphabetische Substitutions-Chiffren verbessern einfache monoalphabetische Chiffren, indem sie verschiedene monoalphabetische Substitutionen verwenden, während man die Klartextnachricht verschlüsselt.

## Bemerkung

**Alle diese Techniken haben die folgenden Merkmale gemeinsam:**

- Es wird ein Satz verwandter monoalphabetischer Substitutionsregeln verwendet.
- Ein Schlüssel bestimmt, welche bestimmte Regel für eine bestimmte Umwandlung gewählt wird.

# Vigenère Chiffre

- Die bekannteste und eine der einfachsten polyalphabetischen Substitutions-Chiffren.
- In diesem Schema besteht die Menge der verwandten monoalphabetischen Substitutionsregeln aus den 26 Caesar-Chiffren mit Verschiebungen von 0 bis 25.
- Jede Chiffre wird durch einen Schlüsselbuchstaben identifiziert, der den Klartextbuchstaben durch den Chiffretextbuchstaben ersetzt.

# Aufbau des Vigenère-Tableaus

- Kopfzeile:  
Klartextbuchstabe
- 1. Spalte:  
Schlüsselbuchstabe
- Tableau:  
Verschlüsselter  
Buchstabe

## Beispiel

Nehmen wir an, der Schlüssel ist "D" und der Klartextbuchstabe sei "b". Dann ist der Chiffretextbuchstabe "E".

/	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

# Beispiel einer Vigenère-Verschlüsselung

- Um eine Nachricht zu verschlüsseln, wird ein Schlüssel benötigt, der so lang ist wie die Nachricht.
- In der Regel ist der Schlüssel ein sich wiederholendes Schlüsselwort.

## Beispiel

Wenn das Schlüsselwort `deceptive` ist, wird die Nachricht „We are discovered save yourself“ wie folgt verschlüsselt:

Schlüssel:   DECEPTIVEDECEPTIVEDECEPTIVE  
Klartext:    wearediscoveredsaveyourself  
Geheimtext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

# Vigenère *Autokey System*

Ein Schlüsselwort wird mit dem Klartext selbst verkettet, um einen laufenden Schlüssel zu erhalten.

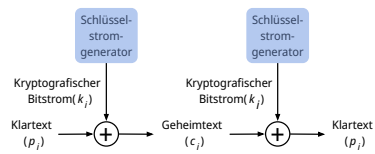
## Beispiel

<b>Schlüssel</b>	DECEPTIVEwearediscoveredsav
<b>Klartext</b>	wearediscoveredsaveyourself
<b>Geheimtext</b>	ZICVTWQNGKZEIIGASXSTSLVVWLA

Auch dieses Verfahren ist anfällig für eine Kryptoanalyse, da der Schlüssel und der Klartext die gleiche Häufigkeitsverteilung der Buchstaben aufweisen und eine statistische Technik angewendet werden kann.



# Vernam Chiffre



# One-Time Pad

- Verbesserung der Vernam-Chiffre, vorgeschlagen von dem Offizier Joseph Mauborgne des Army Signal Corp.
- Verwendung eines Zufallsschlüssels, der so lang wie die Nachricht ist, so dass der Schlüssel nicht wiederholt werden muss.
- Der Schlüssel wird zum Ver- und Entschlüsseln einer einzigen Nachricht verwendet und dann verworfen.
- Jede neue Nachricht erfordert einen neuen Schlüssel mit der gleichen Länge wie die neue Nachricht.
- Das Schema ist nachweislich nicht zu knacken.
  - Erzeugt eine zufällige Ausgabe, die in keinem statistischen Zusammenhang mit dem Klartext steht.
  - Da der Chiffriertext keinerlei Informationen über den Klartext enthält, gibt es keine Möglichkeit, den Code zu knacken.

# Schwierigkeiten von One-Time-Pads

- Das One-Time-Pad bietet vollständige Sicherheit, hat aber in der Praxis zwei grundlegende Schwierigkeiten:

1. Es gibt das praktische Problem der Herstellung großer Mengen von Zufallsschlüsseln.

Jedes stark genutzte System könnte regelmäßig Millionen von zufälligen Zeichen benötigen.

2. Ein „gigantisches“ Schlüsselverteilungsproblem

Für jede zu übermittelnde Nachricht benötigen Sender und Empfänger einen gleich langen Schlüssel.

- Aufgrund dieser Schwierigkeiten ist das One-Time-Pad nur von begrenztem Nutzen; es eignet sich vor allem für Kanäle mit geringer Bandbreite, die eine sehr hohe Sicherheit erfordern.
- Das One-Time-Pad ist das einzige Kryptosystem, das eine perfekte Geheimhaltung bietet.



# Skytale

- Ältestes bekanntes (militärisches) Verschlüsselungsverfahren.
- Vor mehr als 2500 Jahren (vermutlich) von den Spartanern entwickelt.
- Die Verschlüsselung erfolgte mit einem (Holz-)Stab mit einem bestimmten Durchmesser („Schlüssel“) (Skytale).







# Blockchiffre

- Ein Klartextblock wird als Ganzes behandelt und verwendet, um einen gleich langen Chiffretextblock zu erzeugen.
- In der Regel wird eine Blockgröße von 64 (8 Byte) oder 128 Bit (16 Byte) verwendet.
- beide Benutzer teilen sich einen symmetrischen Chiffrierschlüssel
- Viele netzbasierte Anwendungen, die auf symmetrische Verschlüsselung setzen, verwenden Blockchiffren.



# Feistel-Chiffre

Feistel schlug die Verwendung einer Chiffre vor, bei der sich Substitutionen und Permutationen abwechseln.

## Definition

### Substitutionen

Jedes Klartextelement oder jede Gruppe von Elementen wird eindeutig durch ein entsprechendes Chiffretextelement oder eine entsprechende Gruppe von Elementen ersetzt.

## Definition

### Permutation

Bei einer Permutation werden keine Elemente hinzugefügt, gelöscht oder ersetzt, sondern die Reihenfolge, in der die Elemente in einer Folge erscheinen, wird geändert.

# Feistel-Chiffre - Hintergrund

- Hierbei handelt es sich um eine praktische Anwendung eines Vorschlags von Claude Shannon zur Entwicklung einer Chiffre, bei der sich *Konfusions- und Diffusionsfunktionen* abwechseln.
- Dieser Aufbau wird von vielen (Twofish, Blowfish, Serpent, Mars) - teilweise im Einsatz befindlichen - symmetrischen Blockchiffren verwendet.

## **Diffusion und Konfusion**

- Begriffe, die von Claude Shannon eingeführt wurden, um die beiden grundlegenden Bausteine für jedes kryptografische System zu erfassen.
- Shannons Anliegen war es, die auf statistischer Analyse beruhende Kryptoanalyse zu vereiteln.

---

Blowfish ist zum Beispiel die Basis für das Hashingverfahren `bcrypt`, welches für Passworthashing verwendet wird.

# Diffusion

- Die statistische Struktur des Klartextes wird in weitreichende Statistiken des Chiffretextes überführt, d. h. die **statistische Beziehung zwischen Klartext und Chiffretext wird so komplex wie möglich**.
- Dies wird dadurch erreicht, dass jede Klartextziffer(bzw. -zeichen) den Wert vieler Chiffretextziffern (bzw. -zeichen) beeinflusst.  
(„Lawineneffekt“)
- Die Diffusion kann z. B. durch *Permutationen* erreicht werden.

# Konfusion

- Versucht, **die Beziehung zwischen den Statistiken des Chiffriertextes und dem Wert des Chiffrierschlüssels so komplex wie möglich zu gestalten**, d. h. eine einzige Änderung des Chiffrierschlüssels sollte viele Bits des Chiffriertextes beeinflussen.
- Selbst wenn der Angreifer die Statistik des Chiffretextes einigermaßen in den Griff bekommt, ist die Art und Weise, wie der Schlüssel verwendet wurde, um diesen Chiffretext zu erzeugen, so komplex, dass es schwierig ist, den Schlüssel abzuleiten.
- Die Verwirrung kann z. B. durch *Substitutionen* realisiert werden.

# Feistel-Chiffre

## Verschlüsselung und Entschlüsselung

### Legende

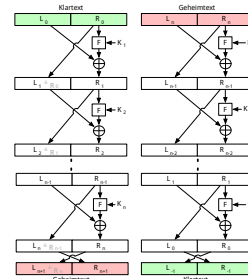
$K_x$  - Schlüssel der  $x$ -ten Runde

$L_{x-1}$  - linke Hälfte des  
Eingabeblocks der  $x$ -ten Runde

$R_{x-1}$  - rechte Hälfte des  
Eingabeblocks der  $x$ -ten Runde

$F$  - Rundenfunktion

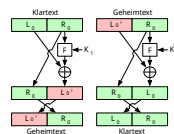
$\oplus$  - XOR-Operation



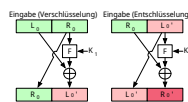
## Notwendigkeit der *Swap* Operation

Die Verwendung des *Swaps* am Ende ist notwendig, damit die Verschlüsselung und Entschlüsselung identisch sind; d. h. derselbe Algorithmus verwendet werden kann.

Zum besseren Verständnis gehen wir im Folgenden davon aus, dass wir nur eine Runde hätten:

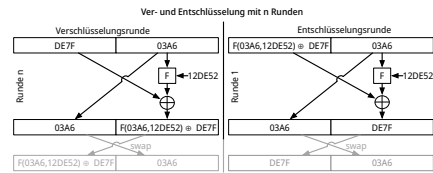


Wird der Swap am Ende nicht durchgeführt, würde die Entschlüsselung nicht funktionieren, wie am folgenden Beispiel mit nur einer Runde zu sehen ist:



Ein alternatives Design wäre es beim Verschlüsseln auf den finalen Tausch zu verzichten und stattdessen beim Entschlüsseln initial einen Tausch durchzuführen — zusätzlich zum finalen Tausch. Dieses Design wird jedoch nicht verwendet, da es die Implementierung komplizierter macht.

# Feistel Chiffre - Beispiel



## Zur Erinnerung


$$[F(03A6, 12DE52) \oplus DE7F] \oplus F(03A6, 12DE52) =$$

$$F(03A6, 12DE52) \oplus F(03A6, 12DE52) \oplus DE7F = DE7F$$

# Entwurfsprinzipien für Blockchiffren - Anzahl der Runden

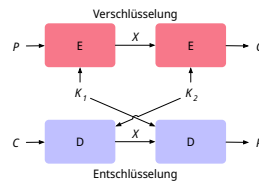
- Je größer die Anzahl der Runden ist, desto schwieriger ist es, eine Kryptoanalyse durchzuführen.
- Im Allgemeinen sollte das Kriterium sein, dass die Anzahl der Runden so gewählt wird, dass bekannte kryptoanalytische Bemühungen mehr Aufwand erfordern als ein einfacher Brute-Force-Schlüsselsuchangriff.
- Hätte DES 15 oder weniger Runden, würde die differentielle Kryptoanalyse weniger Aufwand erfordern als eine Brute-Force-Schlüsselsuche.

# Entwurfsprinzipien für Blockchiffre - Schlüsselableitung

- Bei jeder Feistel-Blockchiffre wird der Hauptschlüssel verwendet, um einen Unterschlüssel für jede Runde zu erzeugen.
- Im Allgemeinen möchten wir die Unterschlüssel so wählen, dass die Schwierigkeit, einzelne Unterschlüssel abzuleiten, und die Schwierigkeit, den Hauptschlüssel wieder zurückzuerhalten, maximiert werden.
- Es wird vorgeschlagen, dass die Schlüsselableitungsfunktion für die Unterschlüssel (  *Key Schedule*) zumindest das **Strenge Lawinenkriterium** und das **Bit-Unabhängigkeitskriterium** für Schlüssel/Ciphertext garantieren sollte.



# Doppelte Verschlüsselung



# Meet-in-the-Middle-Angriff



## Beobachtung

$E(K_2, E(K_1, P)) = E(K_3, P)$  ist nicht gültig.

D. h. die zweifache Anwendung von DES führt zu einer Abbildung, die nicht äquivalent zu einer einfachen DES-Verschlüsselung ist.

Es gilt, dass es keinen Schlüssel  $DES_{K^{1-2}} = DES_{K^1} \circ DES_{K^2}$  gibt. Weiterhin gilt, dass die Teilmenge der durch DES erzeugbaren Permutationen extrem klein ist und nicht abgeschlossen unter Komposition.

Der Meet-in-the-Middle-Algorithmus greift dieses Verfahren an. Er hängt nicht von einer bestimmten Eigenschaft von DES ab, sondern funktioniert gegen jede Blockchiffre.

### ■ Möglicher Known-Plaintext-Angriff:

1. Man berechnet *für alle Schlüssel*  $K_1$  die Chiffretexte  $E(K_1, P)$  und speichert diese.
2. Man berechnet *für alle Schlüssel*  $K_2$  die Klartexte  $D(K_2, C)$ .
3. Man vergleicht die beiden Ergebnisse und prüft, ob es Übereinstimmungen gibt.

Dieser Aufwand ist lediglich doppelt so hoch wie der Aufwand bei einer einfachen Verschlüsselung.

## Warnung

Die zweifache Anwendung einer Blockchiffre ist nicht sinnvoll!

Das Ergebnis ist, dass ein bekannter Klartextangriff gegen Doppel-DES mit einem Aufwand in der Größenordnung von  $2^{56}$  *im Durchschnitt* erfolgreich ist, verglichen mit *durchschnittlich*  $2^{55}$  für einen einfachen DES.

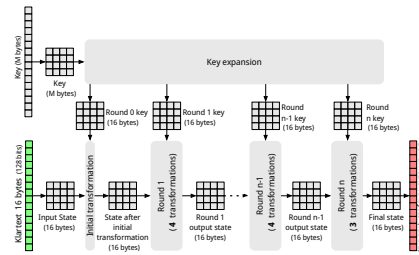
## Advanced Encryption Standard (AES)[2]

Parameter

Schlüsselgröße (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Blockgröße ( <i>Block Size</i> ) (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Anzahl der Runden	10	12	14
Größe des Rundenschlüssels ( <i>RoundKeys</i> ) (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expandierte Schlüsselgröße (words/bytes)	44/176	52/208	60/240

[2] NIST FIPS PUB 197, "Advanced Encryption Standard (AES)"

# AES Verschlüsselungsprozess



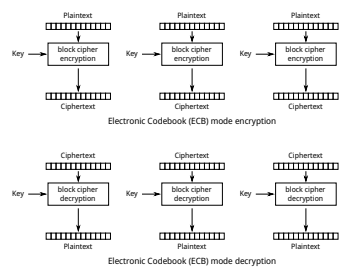
# Betriebsmodi von Blockchiffren

- Eine Technik zur Verbesserung der Wirkung eines kryptografischen Algorithmus oder zur Anpassung des Algorithmus an ein Anwendungsszenario. Insbesondere in Abhängigkeit von der Länge des Klartexts.
- Um eine Blockchiffre in einer Vielzahl von Anwendungen einsetzen zu können, hat das NIST Betriebsmodi definiert.
- Diese Modi sind für die Verwendung mit jeder symmetrischen Blockchiffre vorgesehen, einschließlich AES.

# Betriebsmodi - Übersicht

Modus	Beschreibung	Typische Anwendung
<b>Electronic Codebook (ECB)</b>	Jeder Block von Klartextbits wird unabhängig voneinander mit demselben Schlüssel verschlüsselt.	■ Sichere Übertragung einzelner Werte (z. B. eines Verschlüsselungsschlüssels)
<b>Cipher Block Chaining (CBC)</b>	Die Eingabe für den Verschlüsselungsalgorithmus ist die XOR-Verknüpfung des nächsten Klartextblocks mit dem vorangegangenen Chiffretextblock.	■ Universelle blockorientierte Übertragung ■ Authentifizierung
...	...	...
<b>Counter (CTR)</b>	Jeder Klartextblock wird mit einem verschlüsselten Zähler XOR-verknüpft. Der Zähler wird für jeden nachfolgenden Block erhöht.	■ Blockorientierte Übertragung für allgemeine Zwecke ■ Nützlich für Hochgeschwindigkeitsanforderungen

# Electronic Codebook[3]



[3] Bilder von: **White Timberwolf**

# Padding-Modi in Blockchiffren

## Bemerkung

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

## Ausgewählte Modi

### ■ PKCS#7

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

#### Beispiel

- Nachricht mit 13 Byte → 3 Bytes Padding → 03 03 03
- Nachricht mit Blocklänge (z. B. 16 Byte) → ein kompletter zusätzlicher Block mit  $16 \times 0 \times 10$

### ■ ANSI X.923

Auffüllen mit  $0 \times 00$ , außer dem letzten Byte, das die Anzahl Padding-Bytes angibt.

### ■ ISO/IEC 7816-4

Beginnt mit  $0 \times 80$ , gefolgt von Nullen.

### ■ Zero Padding

#### Achtung!

Füllt mit Nullen ( $0 \times 00$ ). Funktioniert **nur**, wenn die Nachricht nie mit  $0 \times 00$  endet, sonst ist Entschlüsselung mehrdeutig!

## Verhalten bei voller Blockgröße - Zusammenfassung

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

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

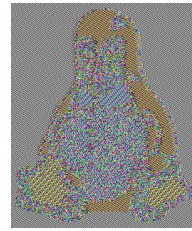
Hinweis: Bei allen Modi wird bei Nachrichten mit dem x-fachen der Blocklänge ein **zusätzlicher** Padding-Block hinzugefügt, um bei der Entschlüsselung korrekt erkennen zu können, ob Padding entfernt werden muss.



# Probleme bei der Verwendung der Verschlüsselung im ECB-Modus

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

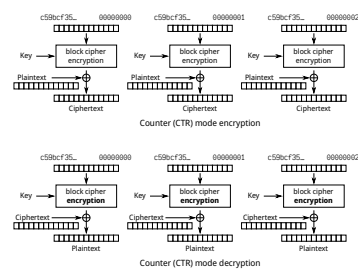
Quelle: <https://github.com/robertdavidgraham/ecb-penguin>



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

- Overhead
- Fehlerbehebung
- Fehlerfortpflanzung
- Streuung
- Sicherheit

# Counter Mode



[4] Bilder von: **White Timberwolf**

## Counter Mode - Vorteile

**Hardware-Effizienz:** kann von der Parallelisierung der Hardware profitieren

**Software-Effizienz:** leicht parallelisierbar in Software

**Vorverarbeitung:** die Verschlüsselung der Zähler

**Zufälliger Zugriff:** der i-te Block des Klartextes/des Chiffretextes kann im Zufallszugriff verarbeitet werden

**Nachweisbare Sicherheit:**


genauso sicher wie die anderen Verfahren

**Einfachheit:** es wird nur der Verschlüsselungsalgorithmus benötigt

**Verwendbarkeit:** Verwandelt eine Blockchiffre in eine Stromchiffre



# Erzeugung und Zufälligkeit von Zufallszahlen

- Eine Reihe von Sicherheitsalgorithmen und -protokollen, die auf Kryptographie basieren, verwenden binäre Zufallszahlen:
  - Schlüsselverteilung und reziproke ( *wechselseitige*) Authentifizierungsverfahren
  - Erzeugung von Sitzungsschlüsseln
  - Generierung von Schlüsseln für den RSA Public-Key-Verschlüsselungsalgorithmus
  - Generierung eines Bitstroms für die symmetrische Stromverschlüsselung

Es gibt zwei unterschiedliche Anforderungen an eine Folge von Zufallszahlen:

- Zufälligkeit
- Unvorhersehbarkeit

# Zufälligkeit

- Die Erzeugung einer Folge von angeblich zufälligen Zahlen, die in einem genau definierten statistischen Sinne zufällig sind, war ein Problem.
- Zwei Kriterien werden verwendet, um zu prüfen, ob eine Zahlenfolge zufällig ist:

**Gleichmäßige Verteilung:**

Die Häufigkeit des Auftretens von Einsen und Nullen sollte ungefähr gleich sein.

**Unabhängigkeit:** Keine Teilsequenz der Folge kann von den anderen abgeleitet werden.

# Unvorhersehbarkeit

Ein Strom von Pseudozufallszahlen sollte zwei Formen der Unvorhersehbarkeit aufweisen:

## 01 Vorwärtsgerichtete Unvorhersehbarkeit

Wenn der Seed unbekannt ist, sollte das nächste erzeugte Bit in der Sequenz trotz Kenntnis der vorherigen Bits in der Sequenz unvorhersehbar sein.

## 02 Rückwärtsgerichtete Unvorhersehbarkeit

- Es sollte nicht möglich sein, den Seed aus der Kenntnis der erzeugten Werte zu bestimmen.
- Es sollte keine Korrelation zwischen einem Seed und einem aus diesem Seed generierten Wert erkennbar sein.
- Jedes Element der Sequenz sollte wie das Ergebnis eines unabhängigen Zufallsereignisses erscheinen, dessen Wahrscheinlichkeit  $1/2$  ist.

### Hinweis

Dieselbe Reihe von Tests für die Zufälligkeit liefert auch einen Test für die Unvorhersehbarkeit: Eine Zufallsfolge hat keine Korrelation mit einem festen Wert (dem Seed).

# Pseudozufallszahlen

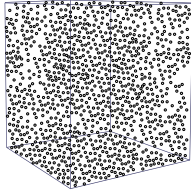
Bei kryptografischen Anwendungen werden in der Regel algorithmische Verfahren zur Erzeugung von Zufallszahlen verwendet.

- Diese Algorithmen sind deterministisch und erzeugen daher Zahlenfolgen, die nicht statistisch zufällig sind.
- Wenn der Algorithmus gut ist, bestehen die resultierenden Sequenzen viele Tests auf Zufälligkeit und werden als Pseudozufallszahlen bezeichnet.

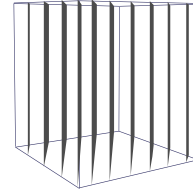


# Visualisierung von (schlechten) Zufallszahlengeneratoren[4]

Erwartete Verteilung von Zufallswerten  
im 3D-Raum.



Verteilung von „zufälligen“ Werten  
eines schlechten RNGs im 3D-Raum.



---

Bei diesem Experiment werden immer drei nacheinander auftretende Werte als Koordinate im 3D-Raum interpretiert. Die erwartete Verteilung ist eine gleichmäßige Verteilung im Raum. Die Verteilung der Werte eines schlechten RNGs ist nicht gleichmäßig und zeigt eine klare Struktur.

---

[5] Zufallszahlengenerator  $\hat{=}$   *Random Number Generator (RNG)*

# Echter Zufallszahlengenerator (TRNG)

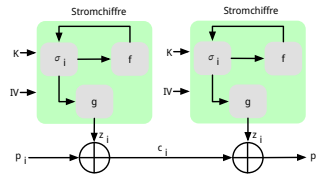
- Nimmt als Eingabe eine Quelle, die effektiv zufällig ist.
- Die Quelle wird als Entropiequelle bezeichnet und stammt aus der physischen Umgebung des Computers:
  - Dazu gehören z. B. Zeitpunkte von Tastenanschlägen, elektrische Aktivität auf der Festplatte, Mausbewegungen und Momentanwerte der Systemuhr.
  - Die Quelle oder eine Kombination von Quellen dient als Eingabe für einen Algorithmus, der eine binäre Zufallsausgabe erzeugt.
- Der TRNG kann einfach die Umwandlung einer analogen Quelle in eine binäre Ausgabe beinhalten.
- Der TRNG kann zusätzliche Verarbeitungsschritte durchführen, um etwaige Verzerrungen in der Quelle auszugleichen.
- Moderne Prozessoren enthalten Instruktionen die echte Zufallszahlen generieren können (sollen).

# Übung

## 3.1. Test auf Zufälligkeit

Test auf Zufälligkeit: Gegeben sei eine Bitfolge, die von einem RNG erzeugt wurde. Was ist das erwartete Ergebnis, wenn man gängige Komprimierungsprogramme (z. B. 7zip, gzip, rar, ...) verwendet, um die Datei zu komprimieren; d. h. welchen Kompressionsgrad erwarten Sie?

# Allgemeine Struktur einer typischen Stromchiffre



Klartext:  $p_i$

Schlüssel:  $K$

Zustand:  $\sigma_i$

Chiffretext:  $c_i$

Initialisierungswert:  
 $IV$

Funktion zur Berechnung des nächsten  
Zustands:  $f$

Schlüsselstrom:  $z_i$

Schlüsselstromfunktion:  $g$

Die Schlüsselstromfunktion  $g$  projiziert den internen Zustand (welcher sehr groß sein kann) auf den nächsten Block/das nächste Byte, das zur Verschlüsselung verwendet wird.

# Überlegungen zum Entwurf von Stromchiffren

## **Die Verschlüsselungssequenz sollte eine große Periode haben:**

Ein Pseudozufallszahlengenerator verwendet eine Funktion, die einen deterministischen Strom von Bits erzeugt, der sich schließlich wiederholt; je länger die Wiederholungsperiode, desto schwieriger wird die Kryptoanalyse.

## **Der Schlüsselstrom sollte die Eigenschaften eines echten Zufallszahlenstroms so gut wie möglich nachbilden:**

Es sollte eine ungefähr gleiche Anzahl von 1en und 0en geben.

Wenn der Schlüsselstrom als ein Strom von Bytes behandelt wird, sollten alle 256 möglichen Byte-Werte ungefähr gleich oft vorkommen.

## **Eine Schlüssellänge von mindestens 128 Bit ist wünschenswert:**

Die Ausgabe des Pseudo-Zufallszahlengenerators ist vom Wert des Eingabeschlüssels abhängig.

Es gelten die gleichen Überlegungen wie für Blockchiffren.

## **Mit einem richtig konzipierten Pseudozufallszahlengenerator kann eine Stromchiffre genauso sicher sein wie eine Blockchiffre mit vergleichbarer Schlüssellänge:**

Ein potenzieller Vorteil ist, dass Stromchiffren, die keine Blockchiffren als Baustein verwenden, in der Regel schneller sind und weit weniger Code benötigen als Blockchiffren.

# Historische Stromchiffre: RC 4

- 1987 von Ron Rivest für RSA Security entwickelt.
- Stromchiffre mit variabler Schlüsselgröße und byteorientierten Operationen, die in Software sehr schnell ausgeführt werden können.
- Basiert auf der Verwendung einer zufälligen Permutation.

## Warnung

In der RC 4-Schlüsselableitungsfunktion wurde eine grundlegende Schwachstelle aufgedeckt, die den Aufwand für die Ermittlung des Schlüssels verringert.

Es wurde gezeigt, dass es möglich ist *wiederholt* verschlüsselte Klartexte wiederherzustellen.

Aufgrund der Schwachstellen hat die IETF RFC 7465 herausgegeben, der die Verwendung von RC4 in TLS verbietet. In seinen TLS-Richtlinien verbietet das NIST ebenfalls die Verwendung von RC4 für Regierungszwecke.

# ChaCha20

- ChaCha20 ist eine Stromchiffre, die von Daniel J. Bernstein entwickelt wurde.
- ChaCha20 ist ein schneller Verschlüsselungsalgorithmus (ohne besondere Hardwareanforderungen).  
Reine Softwareimplementierungen von ChaCha20 sind reinen Softwareimplementierungen von AES in Bezug auf die Geschwindigkeit überlegen.
- ChaCha20 ist im **RFC 8439** spezifiziert.





# Text-basierte Steganografie

## 3.2. Entschlüsseln

Dear Friend ; We know you are interested in receiving cutting-edge announcement . If you are not interested in our publications and wish to be removed from our lists, simply do NOT respond and ignore this mail . This mail is being sent in compliance with Senate bill 1626 ; Title 4 , Section 305 . This is a legitimate business proposal ! Why work for somebody else when you can become rich in 96 months . Have you ever noticed nobody is getting any younger & nobody is getting any younger . Well, now is your chance to capitalize on this ! We will help you decrease perceived waiting time by 170% and use credit cards on your website ! You are guaranteed to succeed because we take all the risk ! But don't believe us . Mrs Anderson of Indiana tried us and says "I was skeptical but it worked for me" . We assure you that we operate within all applicable laws . You will blame yourself forever if you don't order now . Sign up a friend and you'll get a discount of 10% ! Thank-you for your serious consideration of our offer !

Mit Spammimic <https://www.spammimic.com/>, kann die Nachricht extrahiert werden.

# Auswahl anderer Steganographie-Techniken

## ■ **Zeichenmarkierung**

Ausgewählte Buchstaben eines gedruckten oder maschinengeschriebenen Textes werden mit Bleistift überstrichen. Die Markierungen sind nur sichtbar, wenn das Papier schräg in helles Licht gehalten wird.

## ■ **Unsichtbare Tinte**

Es gibt eine Reihe von Substanzen, die zum Schreiben verwendet werden können, aber keine sichtbaren Spuren hinterlassen, solange das Papier nicht erhitzt oder mit einer chemischen Substanz behandelt wird.

## ■ **Nadelstiche**

Kleine Nadelstiche auf ausgewählten Buchstaben sind normalerweise nicht sichtbar, es sei denn, das Papier wird vor ein Licht gehalten.

## ■ **Sehr helle Tinte**

Druckerhersteller drucken winzige Punktmuster in sehr hellen Farben auf die Seiten. Dies erlaubt es Dokumente zu dem Drucker zurückzuverfolgen, auf dem sie gedruckt wurden.

# Steganographie vs. Verschlüsselung

- Steganografie hat eine Reihe von *Nachteilen* im Vergleich zur Verschlüsselung:
  - ! Es erfordert einen hohen Overhead, um relativ wenige Bits an Informationen zu verbergen.
  - ! Sobald das System entdeckt wird, wird es praktisch wertlos.
- Der *Vorteil* der Steganografie:
  - ✓ Sie kann von Parteien eingesetzt werden, die etwas zu verlieren haben, wenn die Tatsache ihrer geheimen Kommunikation (nicht unbedingt der Inhalt) entdeckt wird.
  - ✓ Verschlüsselung kennzeichnet den Verkehr als wichtig oder geheim oder kann den Sender oder Empfänger als jemanden identifizieren, der etwas zu verbergen hat.

## 4. Hashverfahren bzw. Hashfunktionen

---

# Hashfunktionen

- Eine Hashfunktion  $H$  akzeptiert eine beliebig lange Nachricht  $M$  als Eingabe und gibt einen Wert fixer Größe zurück:  $h = H(M)$ .
- Wird oft zur Gewährleistung der Datenintegrität verwendet. Eine Änderung eines beliebigen Bits in  $M$  sollte mit hoher Wahrscheinlichkeit zu einer Änderung des Hashwerts  $h$  führen.
- Kryptographische Hashfunktionen werden für Sicherheitsanwendungen benötigt. Mögliche Anwendungen:
  - Authentifizierung von Nachrichten
  - Digitale Signaturen
  - Speicherung von Passwörtern

# Beispiel: Berechnung von Hashwerten mittels MD5

```
md5("Hello") = 8b1a9953c4611296a827abf8c47804d7
md5("hello") = 5d41402abc4b2a76b9719d911017c592
md5("Dieses Passwort ist wirklich total sicher
    und falls Du es mir nicht glaubst, dann
    tippe es zweimal hintereinander blind
    fehlerfrei ein.")
    = 8fcf22b1f8327e3a005f0cba48dd44c8
```

## Warnung

Die Verwendung von MD5 dient hier lediglich der Illustration. In realen Anwendung sollte MD5 nicht mehr verwendet werden, da es nachgewiesene Schwachstellen aufweist.

# Sicherheitsanforderungen an kryptografische Hashfunktion

## Variable Eingabegröße:

$H$  kann auf einen Block beliebiger Größe angewendet werden.

**Pseudozufälligkeit:** Die Ausgabe von  $H$  erfüllt die Standardtests für Pseudozufälligkeit.

## Einweg Eigenschaft:

Es ist rechnerisch/praktisch nicht machbar für einen gegebenen

Hashwert  $h$  ein  $N$  zu finden so dass gilt:  $H(N) = h$

( *Preimage resistant; one-way property*)


## Schwache Kollisionsresistenz:

Es ist rechnerisch nicht machbar für eine gegebene Nachricht  $M$  eine Nachricht  $N$  zu finden so dass gilt:  $M \neq N$  mit  $H(M) = H(N)$

( *Second preimage resistant; weak collision resistant*)

## Starke Kollisionsresistenz:

Es ist rechnerisch unmöglich ein paar  $(N, M)$  zu finden so dass gilt:  $H(M) = H(N)$ .

( *Collision resistant; strong collision resistant*)

---

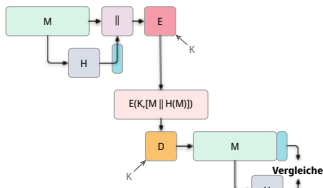
## Hintergrund

Im Deutschen wird auch von Urbild-Angriffen gesprochen. In dem Fall ist *preimage resistance* (d. h. die Einweg Eigenschaft) gleichbedeutend damit, dass man nicht effektiv einen „Erstes-Urbild-Angriff“ durchführen kann. Hierbei ist das Urbild die ursprüngliche Nachricht  $M$ , die *gehasht* wurde.

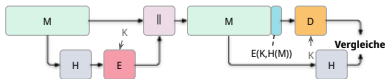
*Second preimage resistance* ist dann gleichbedeutend damit, dass man nicht effektiv einen „Zweites-Urbild-Angriff“ durchführen kann. Es ist nicht möglich zu einer Nachricht  $M$  eine zweite Nachricht  $N$  (d. h. ein zweites Urbild) zu finden, die für eine gegebene Hashfunktion den gleichen Hash aufweist.

# Nachrichtenthauthentifizierung - vereinfacht

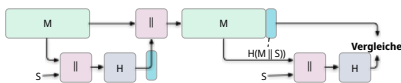
Nachrichten können auf verschiedene Weisen authentifiziert werden, so dass *Person-in-the-Middle-Angriffe*[5] verhindert werden können.



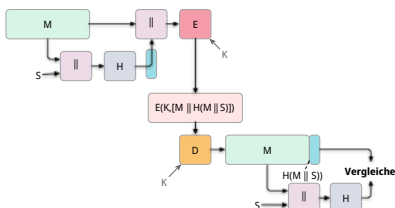
*Garantiert Authentizität und Vertraulichkeit.*



*Garantiert Authentizität, benötigt aber sowohl Hashing als auch Verschlüsselung.*



*Garantiert Authentizität und es wird nur ein Hashalgorithmus benötigt; anfällig für bestimmte Angriffe  
- insbesondere gegen Angriffe mit Längenerweiterung bei Hashverfahren basierend auf  
Merkle-Damgard Konstruktionen.*



*Garantiert Authentizität und Vertraulichkeit.*

---

## Szenarien

Im ersten Szenario wird der Hash an die Nachricht angehängt und als Ganzes verschlüsselt. Wir erhalten Vertraulichkeit und Authentizität.

Im zweiten Szenario wird der Hash der Nachricht berechnet und dann verschlüsselt. Der Empfänger kann den Hash berechnen und mit dem entschlüsselten Hash vergleichen. Wir erhalten Authentizität, aber keine Vertraulichkeit.

Im dritten Szenario wird an die Nachricht ein geteiltes Secret angehängt und alles zusammen gehasht. Die Nachricht wird dann mit dem Ergebnis der vorhergehenden Operation zusammen verschickt.

Im letzten Szenario werden alle Ansätze kombiniert.

### Legende

M:	die Nachricht
H:	die Hashfunktion
E:	der Verschlüsselungsalgorithmus
D:	der Entschlüsselungsalgorithmus



K:	ein geheimer Schlüssel
S:	eine geheime Zeichenkette
:	die Konkatenation von zwei Werten (d. h. das Aneinanderhängen von zwei Werten)

### Hinweis

Bei *Person-in-the-Middle-Angriffen* handelt es sich um einen Fachbegriff und häufig wird zum Beispiel Eve oder Mallory verwendet, um die Person zu bezeichnen, die den Angriff durchführt. Gelegentlich wird auch *Adversary-in-the-Middle* oder früher *Man-in-the-Middle* verwendet.

## Hashes und Message-Digests

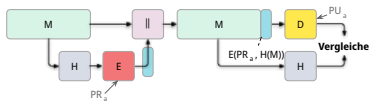
Im allgemeinen Sprachgebrauch wird auch von  *Message Digests* gesprochen.

[6] *Person-in-the-Middle* (in Standards auch *on-path attack*) ist die gender-neutrale Version von *man-in-the-Middle*.

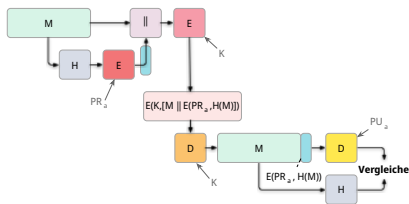
# Digitale Signaturen - vereinfacht

Digitale Signaturen dienen (auch) dem Nachweis der **Authentizität** und **Integrität** einer Nachricht. Darüber hinaus garantieren sie die **Nichtabstreitbarkeit**: Der Absender kann nicht bestreiten, der Urheber der Nachricht zu sein.

Jede Person, die den öffentlichen Schlüssel besitzt, kann die Signatur überprüfen. Nur der Inhaber des zugehörigen privaten Schlüssels ist jedoch in der Lage, die Signatur zu erzeugen.



*Authentizität und Nichtabstreitbarkeit*



*Authentizität, Vertraulichkeit und Nichtabstreitbarkeit*

## Legende

M:	die Nachricht
H:	die Hashfunktion
E:	der Verschlüsselungsalgorithmus
D:	der Entschlüsselungsalgorithmus
$PR_a$ :	der private Schlüssel von a
$PU_a$ :	der öffentliche Schlüssel von a
$  $ :	die Konkatination von zwei Werten (d. h. das Aneinanderhängen von zwei Werten)

# Anforderungen an die Resistenz von Hashfunktionen

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + Digitale Signaturen	✓	✓	✓
Einbruchserkennung und Viruserkennung		✓	
Hash + Symmetrische Verschlüsselung			
Passwortspeicherung	✓		
MAC	✓	✓	✓

## Einbruchserkennung und Viruserkennung - Hintergrund

Bei der Einbruchserkennung und Viruserkennung ist *second preimage* Resistenz erforderlich. Andernfalls könnte ein Angreifer seine Malware so schreiben, dass diese einen Hash wie eine vorhandene gutartige Software hat und so verhindern, dass die Malware auf eine schwarze Liste gesetzt werden kann, ohne den Kollateralschaden, dass auch die gutartige Software fälschlicherweise als Malware erkannt wird.

## Aufwand eines Kollisionsangriffs

Ein Kollisionsangriff erfordert weniger Aufwand als ein *preimage* oder ein *second preimage* Angriff.

Dies wird durch das Geburtstagsparadoxon erklärt. Wählt man Zufallsvariablen aus einer Gleichverteilung im Bereich von 0 bis  $N - 1$ , so übersteigt die Wahrscheinlichkeit, dass ein sich wiederholendes Element gefunden wird, nach  $\sqrt{N}$  (für große  $N$ ) Auswahlen 0,5. Wenn wir also für einen  $m$ -Bit-Hashwert Datenblöcke zufällig auswählen, können wir erwarten, zwei Datenblöcke innerhalb von  $\sqrt{2^m} = 2^{m/2}$  Versuchen zu finden.

### Beispiel

Es ist relativ einfach, ähnliche Meldungen zu erstellen. Wenn ein Text 8 Stellen hat, an denen ein Wort mit einem anderen ausgetauscht werden kann, dann hat man bereits  $2^8$  verschiedene Texte.

Es ist relativ trivial(1), vergleichbare(2) Nachrichten(3) zu schreiben(4). Wenn ein Text 8 Stellen hat, an denen ein Ausdruck(5) mit einem vergleichbaren (6) ausgetauscht werden kann, dann erhält(7) man bereits  $2^8$  verschiedene Dokumente(8).

# Effizienzanforderungen an kryptografische Hashfunktionen

## Effizienz bei der Verwendung für Signaturen und zur Authentifizierung:

Bei der Verwendung zur Nachrichtenauthentifizierung und für digitale Signaturen ist  $H(N)$  für jedes beliebige  $N$  relativ einfach zu berechnen. Dies soll sowohl Hardware- als auch Softwareimplementierungen ermöglichen.

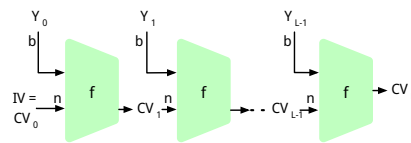
**vs.**

## Brute-Force-Angriffe auf Passwörter erschweren:

Bei der Verwendung für das Hashing von Passwörtern soll es schwierig sein den Hash effizient zu berechnen, selbst auf spezialisierter Hardware (GPUs, ASICs).

# Struktur eines sicheren Hash-Codes

(Vorgeschlagen von Merkle.)



<b>IV:</b>	Initialer Wert (Algorithmus-abhängig)	<b>n:</b>	Länge des Blocks
<b>CV<sub>i</sub>:</b>	Verkettungsvariable	<b>L:</b>	Anzahl der Eingabeblocke
<b>Y<sub>i</sub>:</b>	i-er Eingabeblock	<b>b:</b>	Länge des Eingabeblocks
<b>f:</b>	Kompressionsfunktion		

---

Diese Struktur liegt insbesondere den Hashfunktionen der SHA-2 Familie zugrunde.

# Übung

## 4.1. Irrelevanz von Second-Preimage-Resistenz und Kollisionssicherheit

Warum sind *Second-Preimage-Resistenz* und Kollisionssicherheit von nachgeordneter Relevanz, wenn der Hash-Algorithmus zum Hashing von Passwörtern verwendet wird?

# Hashverfahren: Die SHA-Familie (Secure Hash Algorithms)

- eine Gruppe kryptographischer Hashfunktionen, die von der US-amerikanischen NIST standardisiert wurden.
- Anwendungsziele:
  - Prüfsummenbildung (Integrität)
  - Digitale Signaturen
  - Basis für weitere kryptographische Konstrukte (z. B. HMAC)
- Mitglieder:
  - SHA-1 hat 160 Bit und wird seit 2004 nicht mehr als sicher betrachtet; praktikable Angriffe gibt es seit 2009
  - SHA-2 entwickelt im Jahr 2000 und umfasst 224, 256, 384 und 512 Bit Varianten
  - SHA-3 2015 spezifiziert und basiert auf Keccak

# Zero-Knowledge Protokolle

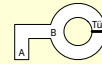
Die Idee ist, dass man jemanden davon überzeugen möchte, dass man eine bestimmte Information hat, ohne diese Information zu offenbaren.

## Beispiel

### Der geheimnisvolle Geheimgang

Peggy möchte Victor überzeugen, dass Sie den Code zur Tür kennt, ohne ihn zu offenbaren.

- Peggy wählt einen der Wege zur Tür, während Victor an der Stelle A steht und darauf wartet, dass Sie bei der Tür ist.
- Sobald Peggy Bescheid gibt, dass Sie an der Tür angekommen ist, geht Victor zu Punkt B und sagt Peggy auf welchem Weg sie zurückkommen soll.
- Kommt sie auf dem falschen Weg zurück, dann kennt sie den Code der Tür (offensichtlich) nicht. Kommt sie auf dem richtigen Weg zurück, könnte es noch immer Zufall gewesen sein mit Wahrscheinlichkeit  $\frac{1}{2}$ .



Wird das Spiel jedoch  $n$  mal gespielt und Peggy kommt immer auf dem richtigen Weg zurück, dann ist die Wahrscheinlichkeit, dass Peggy immer zufällig den richtigen Weg genommen hat  $\frac{1}{2^n}$ .

Viele Zero-Knowledge Protokolle basieren darauf, dass man im Prinzip ein Spiel spielt, das man auch zufällig gewinnen kann. Durch die Wiederholung des Spiels wird die Wahrscheinlichkeit jedoch für permanentes zufälliges Gewinnen sehr schnell sehr klein (exponentiell). Somit kann man für praktische Zwecke hinreichend sicher sein, dass der Beweisführende (im Beispiel Peggy) über das Wissen verfügt, das er vorgibt zu besitzen, wenn er immer gewinnt.

Nach 20 Runden ist die Wahrscheinlichkeit nur noch  $1/2^{20} = 1/1048576$ .

Mit 128 Runden erreicht man ein Sicherheitsniveau, das vergleichbar ist mit anderen kryptographischen Verfahren (AES-128, SHA-256, ...).



## Message Authentication Codes (MACs)

---

### Hinweis

*Message Authentication Codes* könnte ins Deutsche mit Nachrichtenauthentifizierungscodes übersetzt werden, dies ist aber nicht üblich. Im allgemeinen (IT-)Sprachgebrauch wird von *MACs* gesprochen.

# HMAC (Hash-based Message Authentication Code)

- Auch als *keyed-hash message authentication code* bezeichnet.
- Konstruktion:

$$\begin{aligned} \text{HMAC}(K, m) &= H((K' \oplus \text{opad}) || H((K' \oplus \text{ipad}) || m)) \\ K' &= \begin{cases} H(K) & \text{falls } K \text{ größer als die Blockgröße ist} \\ K & \text{andernfalls} \end{cases} \end{aligned}$$

- Standardisiert - sicher gegen Längenerweiterungsangriffe.

---

## Legende

$H$  ist eine kryptografische Hashfunktion.

$m$  ist die Nachricht.

$K$  ist der geheime Schlüssel (*Secret Key*).

$K'$  ist vom Schlüssel  $K$  abgeleiteter Schlüssel mit Blockgröße (ggf. *padded* oder *gehasht*).

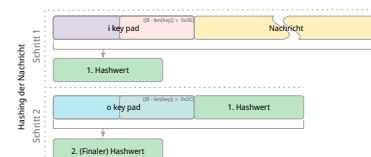
$||$  ist die Konkatenation.

$\oplus$  ist die XOR Operation.

*opad* ist das äußere Padding bestehend aus Wiederholungen von 0x5c in Blockgröße.

*ipad* ist das innere Padding bestehend aus Wiederholungen von 0x36 in Blockgröße.

Abbildung der Schlüssel, die jeweils in die Hashberechnung eingehen. Die Blockgröße sei 8 Bytes und der Schlüssel (Key) sei 16 Bytes.



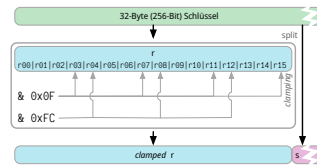
## Padding und Hashing

Im Rahmen der Speicherung von Passwörtern und *Secret Keys* ist die Verwendung von Padding Operationen bzw. das Hashing von Passwörtern, um Eingaben in einer wohl-definierten Länge zu bekommen, üblich. Neben dem hier gesehenen Padding, bei dem 0x00 Werte angefügt werden, ist zum Beispiel auch das einfache Wiederholen des ursprünglichen Wertes, bis man auf die notwendige Länge kommt, ein Ansatz.

Diese Art Padding darf jedoch nicht verwechselt werden mit dem Padding, dass ggf. im Rahmen der Verschlüsselung von Nachrichten notwendig ist, um diese ggf. auf eine bestimmte Blockgröße zu bringen (zum Beispiel bei ECB bzw. CBC Block Mode Operations.)

# MAC: Poly 1305

- Ein MAC Algorithmus für die Einmalauthentifizierung von Nachrichten.
- Entwickelt von Daniel J. Bernstein.
- Basierend auf einem 256-Bit-Schlüssel und einer Nachricht wird ein 128-Bit-Tag berechnet.
- (Insbesondere) In Verbindung mit *ChaCha20* in einer Reihe von Protokollen verwendet.



## Aufteilung des Schlüssels

### "Clamping"

```
1 | clamped_r = r & 0xffffffffc0xffffffffc0xffffffffc0xffffffff # Zahlen
```

### Verarbeitung der Nachricht

- initialisiere den Akkumulator  $a$  mit 0
- die Nachricht wird in Blöcke von 16 Byte aufgeteilt und als *little-endian* Zahl verarbeitet; d. h. ein Block hat 16 Oktette ( $16 \times 8$  Bit)
- Füge dem Block  $n$  ein Bit jenseits der Anzahl der Oktette des aktuellen Blocks hinzu  $\rightarrow n'$  (D. h. im Falle eines 16-Byte-Blocks wird die Zahl  $2^{128}$  addiert und danach haben wir somit eine 17-Byte-Zahl.)
- Addiere  $n'$  aus dem letzten Schritt zum Akkumulator  $a$  und multipliziere mit  $\text{clamped } r$
- Aktualisiere den Akkumulator mit dem Ergebnis *modulo*  $P$  mit  $P = 2^{130} - 5$ :  $a = ((a + n') \times \text{clamped } r) \bmod P$

### Beispiel

```
0000 43 72 79 70 74 6f 67 72 61 70 68 69 63 20 46 6f Cryptographic Fo
0016 72 75 6d 20 52 65 73 65 61 72 63 68 20 47 72 6f rum Research Gro
0032 75 70 up
```

### Schlüssel

```
r = 85 d6 be 78 57 55 6d 33 7f 44 52 fe 42 d5 06 a8
s = 01 03 80 8a fb 0d b2 fd 4a bf f6 af 41 49 f5 1b
```

⚠ Verwendung der (*normalen*) Zahlendarstellung

```
clamped r = 806d5400e52447c036d555408bed685
s = 1bf54941aff6bf4afdb20dfb8a800301
```

### Verarbeitung des ersten Blocks

```
a = 00
n = 6f4620636968706172676f7470797243
n' = 016f4620636968706172676f7470797243
a + n' = 016f4620636968706172676f7470797243
(a + n') × clamped r = b83fe991ca66800489155dcd69e8426ba2779453994ac90ed284034da565ecf
a = ((a + n') × clamped r) mod P = 2c88c77849d64ae9147ddeb88e69c83fc
```

## Bemerkung

### Berechnung für den ersten Block in Python

```
1 a = 0x00 # initial
2 a = a + n'
3 a = 0x016f4620636968706172676f7470797243 # 0x00 + n' = n'
4 a *= 0x806d5400e52447c036d555408bed685
5 print("((a+n') × clamped r)=", hex(a))
6 print("((a+n') × clamped r) mod P=", hex(a % 0x3fffffffffffffffffffffffffffffffb))
```

Verarbeitung des letzten Blocks mit 2 Bytes

$a =$	<code>2d8adaf23b0337fa7ccfb4ea344b30de</code>
$n =$	<code>7075</code>
$n' =$	<code>017075</code>
$a + n' =$	<code>2d8adaf23b0337fa7ccfb4ea344ca153</code>
$(a + n') \times \text{clamped } r =$	<code>16d8e08a0f3fe1de4fe4a15486aca7a270a29f1e6c849221e4a6798b8e45321f</code>
$a = ((a + n') \times \text{clamped } r) \bmod P =$	<code>28d31b7caf946c77c8844335369d03a7</code>

Abschluss

Addiere auf den Wert des Akkumulators  $a$  den Wert  $s$ .

Somit ist der Tag  $t = a + s = 2a927010caf8b2bc2c6365130c11d06a8$  (als Zahl).

Davon werden die *least-significant 128 Bit* serialisiert und verwendet.

a8 06 1d c1 30 51 36 c6 c2 2b 8b af 0c 01 27 a9

## Serialisierung in Python

```
1 from binascii import hexlify
2
3 t = 0x2a927010caf8b2bc2c6365130c11d06a8
4 hexlify(t.to_bytes(17, byteorder="little")[0:16])
```

## Hinweise

- In dieser Diskussion betrachten wir jeden Block der Nachricht als „große Zahl“.
- $P = 2^{130} - 5 = 3fffffffffffffffffffffffffffffffffffffb$
- Dadurch, dass wir den Block als Zahl in *little-endian* Reihenfolge interpretieren, ist das Hinzufügen des Bits jenseits der Anzahl der Oktette gleichbedeutend damit, dass wir den Wert 0x01 am Ende des Blocks hinzufügen.

## 5. Public-Key-Kryptographie

---


# Terminologie bzgl. asymmetrischer Verschlüsselung

- Asymmetrische Schlüssel
  - Public-Key-Zertifikat
  - Public-Key (asymmetrischer) kryptografischer Algorithmus
  - Public-Key-Infrastruktur (PKI)
- 

## Asymmetrische Schlüssel:

Zwei zusammengehörige Schlüssel, ein öffentlicher und ein privater Schlüssel, die zur Durchführung komplementärer Operationen verwendet werden, z. B. Ver- und Entschlüsselung oder Signaturerstellung und Signaturprüfung.

## Public-Key-Zertifikat:

Ein digitales Dokument, das mit dem privaten Schlüssel einer Zertifizierungsstelle ( *Certification Authority*) ausgestellt und digital signiert wird und den Namen eines Teilnehmers an einen öffentlichen Schlüssel bindet. Das Zertifikat gibt an, dass der im Zertifikat genannte Teilnehmer die alleinige Kontrolle und den Zugriff auf den entsprechenden privaten Schlüssel hat.

## Public-Key (asymmetrischer) kryptografischer Algorithmus:

Ein kryptografischer Algorithmus, der zwei zusammengehörige Schlüssel verwendet, einen öffentlichen und einen privaten Schlüssel. Die beiden Schlüssel haben die Eigenschaft, dass die Ableitung des privaten Schlüssels aus dem öffentlichen Schlüssel rechnerisch nicht machbar ist bzw. sein sollte (vgl. Quantenkryptografie).

## Public-Key-Infrastruktur (PKI):

Eine Reihe von Richtlinien, Prozessen, Serverplattformen, Software und Workstations, die für die Verwaltung von Zertifikaten und öffentlich-privaten Schlüsselpaaren verwendet werden, einschließlich der Möglichkeit, Public-Key-Zertifikate auszustellen, zu pflegen und zu widerrufen.

# Prinzipien von Public-Key-Kryptosystemen

- Das Konzept der *Public-Key-Kryptographie* (d. h. der Kryptografie mit öffentlichen Schlüsseln) entstand aus dem Versuch, zwei der schwierigsten Probleme im Zusammenhang mit der symmetrischen Verschlüsselung zu lösen:

## Schlüsselverteilung

Wie kann man generell sicher kommunizieren, ohne einem "Key Distribution Center" (KDC) seinen Schlüssel anvertrauen zu müssen?

## Digitale Signaturen

Wie kann man überprüfen, ob eine Nachricht unversehrt vom angegebenen Absender stammt?

---

KDC = Key Distribution Center



# Prinzipien von Public-Key-Kryptosystemen



**Whitfield Diffie** und **Martin Hellman** von der Stanford University erzielten 1976 einen Durchbruch, indem sie eine Methode entwickelten, die beide Probleme löste und sich radikal von allen bisherigen Ansätzen der Kryptografie unterschied.

# Bestandteile von Public-Key-Kryptosystemen

## **Klartext** (📄 *Plaintext*):

Die lesbare Nachricht oder Daten, die dem Algorithmus als Eingabe dienen.

## **Verschlüsselungsalgorithmus:**

Führt verschiedene Umwandlungen des Klartextes durch.

## **Öffentlicher Schlüssel:**

Wird für *Verschlüsselung* oder *Entschlüsselung* verwendet.

**Privater Schlüssel:** Verwendet für *Verschlüsselung* oder *Entschlüsselung*.

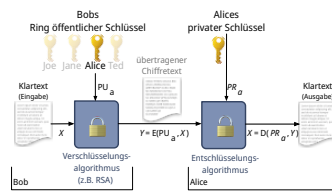
## **Chiffretext** (📄 *Ciphertext*):

Die verschlüsselte Nachricht, die als Ausgabe produziert wird.

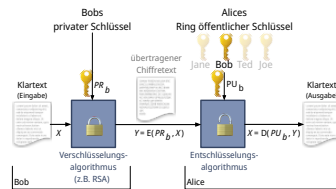
## **Entschlüsselungsalgorithmus:**

Nimmt den Geheimtext und den passenden Schlüssel entgegen und erzeugt den ursprünglichen Klartext.

# Verschlüsselung mit öffentlichem Schlüssel



# Verschlüsselung mit privatem Schlüssel



# Konventionelle und Public-Key-Verschlüsselung

## Konventionelle Verschlüsselung

### Benötigt zur Anwendung

1. Es wird derselbe Algorithmus mit demselben Schlüssel für die Ver- und Entschlüsselung verwendet.
2. Der Sender und der Empfänger müssen den Algorithmus und den Schlüssel kennen bzw. besitzen.

### Notwendig für die Sicherheit

1. Der Schlüssel muss geheim gehalten werden.
2. Es muss unmöglich oder zumindest unpraktisch sein, eine Nachricht zu entschlüsseln, wenn der Schlüssel geheim gehalten wird.
3. Die Kenntnis des Algorithmus und von (ggf. vielen) Geheimtexten ist nicht ausreichend, um den Schlüssel zu ermitteln.

## Public-Key Verschlüsselung

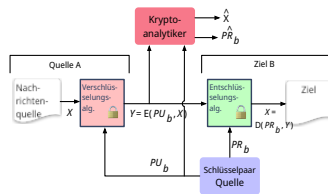
### Benötigt zur Anwendung

1. Zwei Algorithmen: je einer für die Ver-/Entschlüsselung. Weiterhin ein Paar von Schlüsseln; je einer für die Ver-/Entschlüsselung.
2. Der Absender und der Empfänger müssen jeweils einen der passenden Schlüssel besitzen (nicht den gleichen).

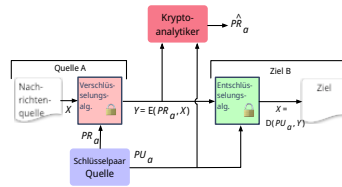
### Notwendig für die Sicherheit

1. Einer der Schlüssel muss geheim bleiben.
2. Es muss unmöglich sein, eine Nachricht zu entschlüsseln, wenn ein Schlüssel geheim gehalten wird.
3. Die Kenntnis des Algorithmus und eines Schlüssels sowie von Geheimtexten ist nicht ausreichend, um den anderen Schlüssel zu ermitteln.

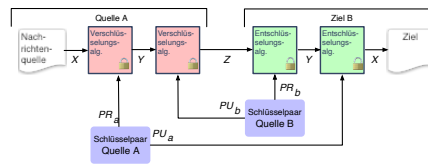
# Public-Key-Kryptosystem: Vertraulichkeit



# Public-Key-Kryptosystem: Authentifizierung



# Public-Key-Kryptosystem: Authentifizierung und Geheimhaltung





# Anwendungen für Public-Key-Kryptosysteme

Kryptosysteme mit öffentlichen Schlüsseln lassen sich in drei Kategorien einteilen:

1. *Ver-/Entschlüsselung*: Der Absender verschlüsselt eine Nachricht mit dem öffentlichen Schlüssel des Empfängers.
2. *Digitale Unterschriften*: Der Absender „unterschreibt“ eine Nachricht mit seinem privaten Schlüssel.
3. *Schlüsselaustausch*: Zwei Seiten arbeiten zusammen, um einen Sitzungsschlüssel (d. h. einen symmetrischen Schlüssel) auszutauschen.

Einige Algorithmen eignen sich für alle drei Anwendungen, während andere nur für eine oder zwei verwendet werden können:

Algorithmus	Ver-/ Entschlüsselung	Digitale Signaturen	Schlüssel- austausch
RSA	✓	✓	✓
Elliptic Curve	✓	✓	✓
Diffie-Hellman	✗	✗	✓
DSS[6]	✗	✓	✗

[7] DSS = Digital Signature Standard\*, entwickelt von der NSA (National Security Agency)

# Anforderungen an Public-Key-Algorithmen

- Für eine Partei  $B$  ist es rechnerisch einfach, ein Schlüsselpaar (bestehend aus öffentlicher Schlüssel  $PU_b$  und privater Schlüssel  $PR_b$ ) zu erzeugen.
- Für einen Absender  $A$  ist es rechnerisch einfach, bei Kenntnis des öffentlichen Schlüssels von  $B$  und der zu verschlüsselnden Nachricht den entsprechenden Chiffretext zu erzeugen.
- Für den Empfänger  $B$  ist es rechnerisch einfach, den resultierenden Chiffretext mit Hilfe des privaten Schlüssels zu entschlüsseln, um die ursprüngliche Nachricht wiederherzustellen.
- Für einen Angreifer, der den öffentlichen Schlüssel kennt, ist es *rechnerisch unmöglich*, den privaten Schlüssel zu ermitteln.
- Für einen Angreifer, der den öffentlichen Schlüssel und einen Chiffretext kennt, ist es *rechnerisch unmöglich*, die ursprüngliche Nachricht wiederherzustellen.
- Die beiden Schlüssel können in beliebiger Reihenfolge verwendet werden.

# Anforderungen an Public-Key-Algorithmen

- Benötigt wird eine Falltürfunktion (🚪 *Trapdoor one-way function*)

Eine Einwegfunktion ist im Allgemeinen eine Funktion, bei der jeder Funktionswert eine eindeutige Umkehrung hat, wobei die *Berechnung der Funktion einfach* ist, während die *Bestimmung der Umkehrfunktion praktisch undurchführbar* ist.

- $Y = f(X)$  einfach
- $X = f^{-1}(Y)$  „unmöglich“
- Eine Einwegfunktion mit Falltür ist eine Familie invertierbarer Funktionen  $f_k$ , für die gilt:
  - $Y = f_k(X)$  einfach, wenn  $k$  und  $X$  bekannt sind.
  - $X = f_k^{-1}(Y)$  einfach, wenn  $k$  und  $Y$  bekannt sind.
  - $X = f_k^{-1}(Y)$  unmöglich, wenn  $Y$  bekannt ist, aber  $k$  nicht.
- Ein praktisches Public-Key-Verfahren hängt von einer geeigneten Trapdoor-Einwegfunktion ab.

---

Ein Falltürfunktion lässt sich nicht trivial umkehren; bzw. die Umkehrung erfordert spezielle (weitergehende) Informationen; d. h. die Falltür.

# Rivest-Shamir-Adleman (RSA) Algorithm

- Entwickelt 1977 am MIT von Ron Rivest, Adi Shamir und Len Adleman.
- Universeller Ansatz zur Verschlüsselung mit öffentlichen Schlüsseln.
- Ist eine Chiffre, bei der Klartext und Chiffretext ganze Zahlen zwischen 0 und  $n - 1$  für ein bestimmtes  $n$  sind.
- Eine typische Größe für  $n$  waren 1024 Bits oder 309 Dezimalziffern.

Solch kleine Zahlen werden heute als äußerst unsicher angesehen, insbesondere angesichts der bevorstehenden Quantencomputer und der Entwicklung von Quantenalgorithmen (vgl. [Shors Algorithmus \(1994\)](#)), die Zahlen effizient faktorisieren können, wenn genügend QBits in hinreichender Qualität[7] zur Verfügung stehen.

---

- [8] Aktuell sind Quantencomputer nicht in der Lage, die für RSA verwendeten Schlüssel zu brechen und es ist auch (noch) nicht geklärt ob die aktuellen Technologien entsprechend skaliert werden können. Es besteht aber die durchaus ernst zu nehmende Möglichkeit!

# RSA Algorithmus

- RSA verwendet einen Ausdruck mit Exponentialen
- Der Klartext wird in Blöcken verschlüsselt, wobei jeder Block einen Binärwert hat, der kleiner als eine bestimmte Zahl  $n$  ist[8].
- Die Ver- und Entschlüsselung erfolgt für einen Klartextblock  $M$  und einen Chiffretextblock  $C$  in der folgenden Form:

$$C = M^e \bmod n \quad M = C^d \bmod n \quad (M^e)^d \bmod n = M^{ed} \bmod n$$

- Sowohl der Sender als auch der Empfänger müssen den Wert von  $n$  kennen.
- Der Absender kennt den Wert von  $e$ , und nur der Empfänger kennt den Wert von  $d$
- Dies ist ein Public-Key-Verschlüsselungsalgorithmus mit dem öffentlichen Schlüssel  $PU = \{e, n\}$  und dem privaten Schlüssel  $PR = \{d, n\}$ .

---

$$M = C^d \bmod n \Rightarrow M = (M^e \bmod n)^d \bmod n = (M^e)^d \bmod n$$

---

- [9] Basierend auf der Zahl  $n$  ergibt sich die maximale Größe des Blocks in Bit. Sei, hypothetisch,  $n = 4.294.967.296 + 1$ , dann kann der Block maximal 32 Bit groß sein ( $2^{32} = 4.294.967.296$ ).

# The RSA Algorithm

## Schlüsselgenerierung von Alice

Wähle $p, q$	$p$ und $q$ beide prim, $p \neq q$
Berechne $n$	$n = p \times q$
Berechne $\phi(n)$	$\phi(n) = (p - 1)(q - 1)$
Wähle $e$	$\text{GGT}(\phi(n), e) = 1; \quad 1 < e < \phi(n)$
Berechne $d$	$d \equiv e^{-1} \pmod{\phi(n)} \Leftrightarrow ed \pmod{\phi(n)} = 1$
Public-Key	$PU = \{e, n\}$
Private-Key	$PR = \{d, n\}$

## Verschlüsselung von Bob mit Alices öffentlichen Schlüssel

Klartext  $M < n$

Chiffretext  $C = M^e \pmod{n}$

## Entschlüsselung von Alice mit ihrem privaten Schlüssel

Chiffretext  $C$

Klartext  $M = C^d \pmod{n}$

## Beispiel für den RSA-Algorithmus

p und q:  $p = 11; \quad q = 17; \quad n = 187 \quad (\phi(n) = 10 \times 16 = 160)$

Klartext: 88

Verschlüsselung:  $PU = \{e = 7, n = 187\}$ :  
 $88^7 \bmod 187 = 11 = C$

Entschlüsselung:  $PR = \{d = 23, n = 187\}$ :  
 $11^{23} \bmod 187 = 88 = P$

Alternativer Exponent:

$$e = 137 \Rightarrow d = 153$$
$$88^{137} \bmod 187 = 99 = C \quad 99^{153} \bmod 187 = 88$$

# Die Sicherheit von RSA - Fünf mögliche Ansätze für einen Angriff

**Brute-Force:** Dabei werden alle möglichen privaten Schlüssel ausprobiert.

**Mathematische Angriffe:**

Es gibt mehrere Ansätze, die vom Aufwand her alle dem Faktorisieren des Produkts aus zwei Primzahlen entsprechen.

**Zeitliche Angriffe:** Diese hängen von der Laufzeit des Entschlüsselungsalgorithmus ab.

**Hardware-Fehler-basierter Angriff:**

Hier geht es darum, Hardware-Fehler in den Prozessor zu induzieren, der digitale Signaturen erzeugt.

**Gewählte Chiffretext-Angriffe:**

Ziel ist es Eigenschaften des RSA-Algorithmus auszunutzen.

**Quantum-Computing:**

Shor's Algorithmus ist ein Algorithmus, der in polynomieller Zeit die Faktorisierung von Zahlen mit einer bestimmten Anzahl von Bits erreicht. Dies bedeutet, dass Quantum-Computing die Sicherheit von RSA-Systemen bedroht, da sie die Faktorisierung von großen Zahlen effizienter lösen kann als klassische Computer.



## 6. Netzwerksicherheit



# TCP Grundlagen


- Protokoll basierend auf IP
- verbindungsorientierte Kommunikation zweier Rechner im Internet zuverlässig und geordnet:
  - Verwerfen von Duplikaten und fehlerhaft übertragener Pakete
  - automatisches Wiederversenden fehlender Pakete
  - Nachrichtenpuffer: Daten werden in korrekter Reihenfolge an Applikation zugestellt
- Verbindungsaufbau immer zwischen zwei Sockets (Socket-Adresse: IP Adresse und 16 Bit-Port-Nummer)

# Aufbau einer TCP Verbindung

Dreifacher Handshake:

---

## Terminologie:

SYN:  *synchronize (session establishment)*

ACK:  *acknowledge*

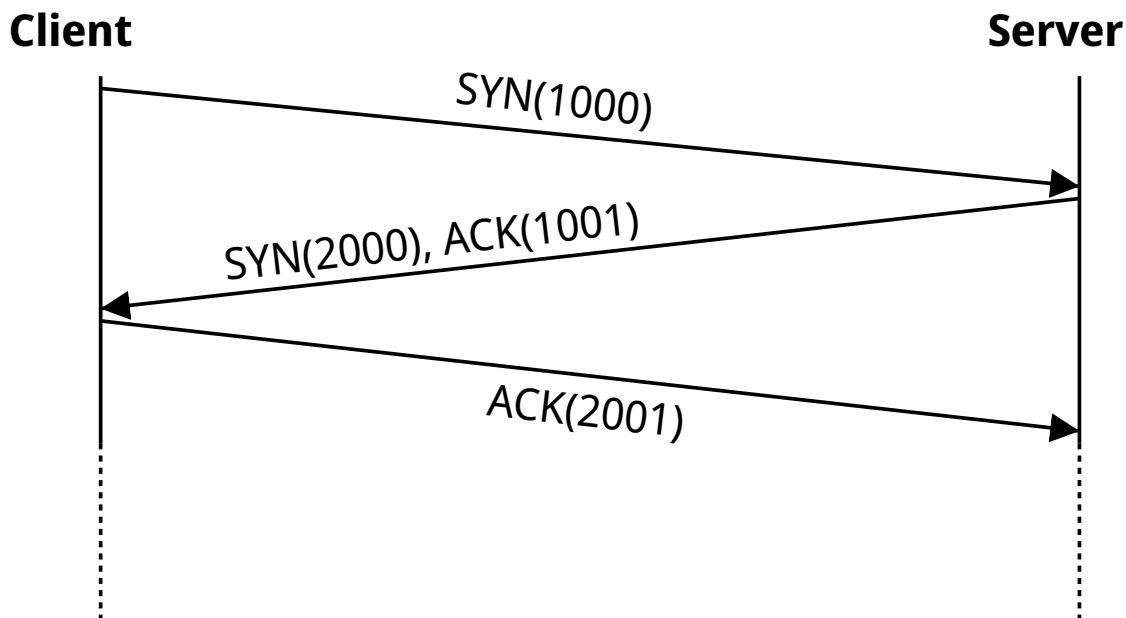
RST:  *reset*

## Verbindungsaufbau - Ablauf:

1. Client sendet SYN Paket mit initialer Sequenznummer (hier) 1000 an den Server.
2. Server sendet ein SYN-ACK Paket mit seiner initialen Sequenznummer (hier) 2000 und ein ACK mit der Sequenznummer 1001 (initiale Sequenznummer des Clients +1) an den Client
3. Client sendet ein ACK Paket mit Sequenznummer 2001 (initiale Sequenznummer des Servers +1) an den Server; danach ist die Verbindung aufgebaut.

Das Betriebssystem sollte die initialen Sequenznummern zufällig wählen, so dass ein Angreifer diese nicht leicht vorhersagen kann. Beide Seiten haben eigene Sequenznummern, die unabhängig voneinander sind.

Bei einer laufenden Verbindung werden die Sequenznummern inkrementiert und es ist nicht (mehr) erkennbar wer die Verbindung aufgebaut hat.



# Ports bei TCP

- Port-Nummern werden für die Kommunikation zwischen zwei Diensten/Prozessen verwendet
- Ports sind 16 Bit Zahlen (0-65535)
- (Unix) Ports < 1024 sind privilegiert (nur root kann diese öffnen)
- einige Port-Nummern sind Standarddiensten zugeordnet

## Port-Nummern einiger Standarddienste [9]

**Ungeschützte Dienste** (Kommunikation findet ohne Verschlüsselung statt.)

Protokoll	Dienst	Portnummer
ftp	Dateitransfer	21
smtp	Simple Mail Transfer Protocol	25
dns	Domain Name System	53
http	Hypertext Transfer Protocol	80
login	Login auf entfernte Rechner	513

**Geschützte Dienste** (Die Kommunikation ist verschlüsselt.)

Protokoll	Dienst	Portnummer
ssh	Secure Shell	22
https	HTTP über Secure Socket Layer	443
smtps	SMTP über Secure Socket Layer	465
imaps	IMAP über Secure Socket Layer	993
pop3s	POP3 über Secure Socket Layer	995

[10]Port numbers assigned by IANA

# Angriffe auf TCP - Motivation

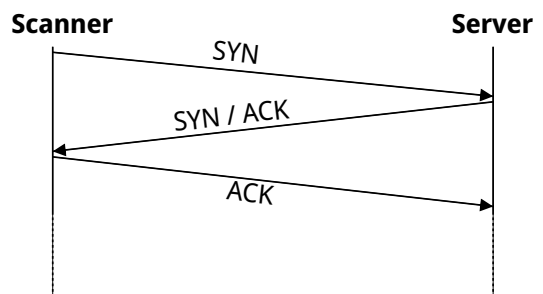
- Netzwerkprogrammierung mit TCP ist relativ komfortabel.
- Viele Dienste sind mit TCP implementiert.
- Das Auffinden von angreifbaren Diensten kann mit Hilfe von Port Scans systematisch erfolgen.

Server haben heutzutage im Allgemeinen alle nicht verwendeten Dienste geschlossen.

# Port Scans: TCP Connect Scan

## Vorgehen

Aufbau vollständiger Verbindungen  
zu allen bzw. zu ausgewählten Ports.



## Bewertung

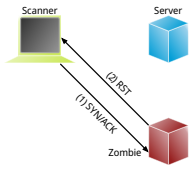
- simpelster Port Scan
- große Entdeckungsgefahr (Scan selbst ist kein Angriff)



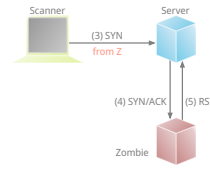
# Port Scans: Idle Scan [10]

Unter Verwendung eines sog. Zombies kann ein Port Scan durchgeführt werden, bei dem der Scanner nicht identifiziert werden kann:

Sondiere IP ID des Zombies:



Starte Scan:



**Zombie:**

ein Rechner (Computer, Drucker oder anderes IoT Gerät) im Internet *möglichst ohne eigenen Netzverkehr* und mit *altem* Betriebssystem, bei dem die IP ID in vorhersehbarer Weise inkrementiert wird. (Bei modernen Betriebssystemen ist die IP ID zufällig, **konstant** oder sogar `null`.)

**Grundlegende Idee:**

Der Zombie sendet ein RST Paket zurück, da er kein SYN gesendet hat und kein SYN/ACK erwartet. Dadurch erfährt der Angreifer die aktuelle IP ID des Zombies. Über diesen Seitenkanal - d. h. die Veränderung der IP ID des Zombies - kann der Angreifer nun den Zustand des Ports auf dem Zielrechner ermitteln.

## Hinweis

Sollte ein Intrusion Detection System vorhanden sein, so wird dieses den Zombie als Angreifer identifizieren.

## Hintergrund - IP ID

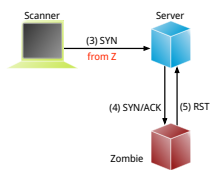
Das Feld *IP Identifikation (IP ID)* dient der Identifizierung einer Gruppe von Fragmenten eines einzelnen IP-Datagramms.

By Michel Bakni - Postel, J. (September 1981) RFC 791, IP Protocol, DARPA Internet Program Protocol Specification, p. 1 DOI: 10.17487/RFC0791., CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=79949694>

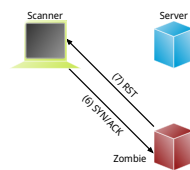
[11]NMap Book

# Port Scans: Idle Scan

## Starte Scan:



## Sondiere IP ID des Zombies:



- Angreifer sendet SYN/ACK Paket an Zombie
- der Zombie antwortet mit RST und enthüllt seine IP ID (IP Fragment Identification Number).
- Angreifer sendet SYN („mit IP vom Zombie“) an Port des Servers:

**[Port offen]** Der Zielrechner antwortet mit SYN/ACK an den Zombie, wenn der Port offen ist. Der Zombie antwortet darauf mit RST an den Server, da er kein SYN gesendet hat und kein SYN/ACK erwartet und *erhöht seine IP ID*.

**[Port geschlossen]** Der Zielrechner antwortet mit RST an den Zombie, wenn der Port geschlossen ist. Dies wird vom Zombie ignoriert. - Der Angreifer sendet wieder ein SYN/ACK an den Zombie, um die IP ID zu erfahren.

Mit einem IDLE Scan kann nicht unterschieden werden, ob der Port geschlossen oder gefiltert ist.

# Port Scans mit nmap

- alle Arten von Port-Scans möglich
- auch OS fingerprinting
- u. U. sogar Ermittlung der Versionsnummern von Diensten

```
$ nmap 192.168.178.121 -Pn
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-14 13:16 PST
Nmap scan report for Michaels-MacBook-Pro (192.168.178.121)
Host is up (0.0056s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
445/tcp    open  microsoft-ds
5000/tcp   open  upnp
7000/tcp   open  afs3-fileserver
```

---

## OS-Fingerprinting

Beim OS-Fingerprinting werden Datenpakete analysiert, die aus einem Netzwerk stammen, um Informationen für spätere Angriffe zu gewinnen. Durch die Erkennung des Betriebssystems, mit dem ein Netzwerk arbeitet, haben Hacker es leichter, Schwachstellen zu finden und auszunutzen. OS-Fingerprinting kann auch Konfigurationsattribute von entfernten Geräten sammeln. Diese Art von Aufklärungsangriff ist in der Regel (einer) der erste(n) Schritt(e).

Es gibt zwei Arten von OS-Fingerprinting: (1) Aktiv und (2) passiv.

1. Bei einem aktiven OS-Fingerprinting-Versuch senden die Angreifer ein Paket an das Zielsystem und warten auf eine Antwort, um den Inhalt des TCP-Pakets zu analysieren.
2. Bei einem passiven Versuch agieren die Angreifer eher als "Schnüffler", der keine absichtlichen Änderungen oder Aktionen im Netzwerk vornimmt. Passives OS-Fingerprinting ist ein unauffälligerer, aber wesentlich langsamerer Prozess.

# Denial-of-Service (DoS) Angriffe

Ziel des Angreifers: Lahmlegen eines Dienstes oder des ganzen Systems ...

- durch Ausnutzen von Schwachstellen (📁 *vulnerabilities*) wie z. B. Buffer Overflows
- durch Generierung von Überlast (Ausschöpfen von RAM, CPU, Netzwerkbandbreite, ...)

## Beispiel

### Ping-of-Death

(Historisch: aus dem Jahr 1997)

Ein `ping` (vgl. Internet Control Message Protocol (ICMP)) verwendet üblicherweise kleine Nachrichten, aber die verwendete Länge ist einstellbar.

Falls die Länge zu groß ist  $\Rightarrow$  Buffer Overflow  $\Rightarrow$  Systemabsturz!

Variante: mittels Fragmentierung ließen sich generell übergroße IP-Pakete (>65,536 Byte) erstellen.

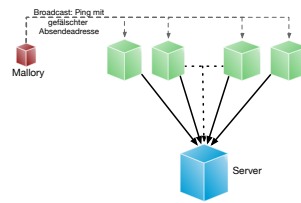
# Denial-of-Service: SYN-flooding Angriff

- Angriff auf Design
- Angreifer sendet eine Verbindungsaufbauanforderung (gesetztes SYN-Flag) an Zielmaschine
- Server generiert eine halboffene TCP-Verbindung
- Angreifer wiederholt in schneller Folge dieses erste Paket zum Verbindungsaufbau
  - ⇒ vollständiges Füllen der internen Systemtabelle
  - ⇒ Anfragen normaler Benutzer werden zurückgewiesen
- Angreifer verwendet i. Allg. IP-Spoofing weswegen Firewalls wirkungslos sind.
- Abwehr: SYN-Cookies

# Distributed Denial-of-Service (DDoS) Angriff

Opfer wird von sehr vielen Angreifern mit Nachrichten überflutet.

Ein Beispiel: Smurf-Angriff:



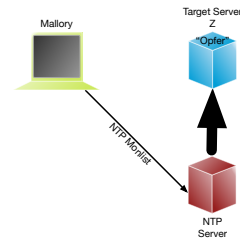
# Distributed Denial-of-Service (DDoS) Angriff

- Bot-Netze (Botnetze) werden verwendet, um DDoS-Angriffe durchzuführen.
- Bot-Netze können viele 10.000 Rechner umfassen.
- IoT Geräte sind besonders beliebt (z. B. IP-Kameras, Smart-TVs, Smart-Home Geräte, ...), da diese oft nicht ausreichend geschützt sind und trotzdem permanent mit dem Internet verbunden sind.
- Beliebte Ziele:
  - Onlinespieleserver
  - Banking-Portale
  - politische Webseiten
- Firewalls und Intrusion Detection Systeme sind meist wirkungslos, da die Angriffe von vielen verschiedenen IP-Adressen kommen.

# Distributed-Reflected-Denial-of-Service Angriff

## Idee eines (DRDoS) Angriffs

- Es wird eine Anfrage an einen Server gesendet, die eine große Antwort auslöst.



-----

Z. B. hat(te) der NTP Monlist Befehl eine Antwort, die ca. 200 Fach größer ist als die Anfrage!

- Mittels IP-Spoofing wird die IP-Adresse des Opfers als Absenderadresse verwendet.
- Es werden insbesondere Dienste basierend auf UDP verwendet, da hier keine Verbindung aufgebaut werden muss.
- Nehmen einen signifikanten Teil aller DDoS-Angriffe ein.
- Die Tatsache, dass die Sender legitime Server sind, erschwert die Abwehr.
- *Egress Filtering* kann helfen, die Verwendung von IP-Spoofing zu verhindern.

-----

Bereits im Jahr 2018 wurde ein Angriff mit einer Bandbreite von 1,7 TBit/s beobachtet.

**Egress Filtering:** Der Router verwirft alle Pakete, die eine Absenderadresse verwenden, die nicht aus dem eigenen Netzwerk stammt.



# Distributed Denial-of-Service (DDoS) Angriffe

*[...] Google's DDoS Response Team has observed the trend that distributed denial-of-service (DDoS) attacks are **increasing exponentially in size**. Last year, we blocked the largest DDoS attack recorded at the time. This August [2023], we stopped an even larger DDoS attack — 7½ times larger — that also used new techniques to try to disrupt websites and Internet services.*

*This new series of DDoS attacks reached **a peak of 398 million requests per second (rps)**, and relied on a novel HTTP/2 “Rapid Reset” technique based on stream multiplexing that has affected multiple Internet infrastructure companies. By contrast, last year's largest-recorded DDoS attack peaked at 46 million rps.*

*—Okt. 2023 - DDoS Attack with 398 Million RPS*

*Cloudflare hat Mitte Mai den "größten jemals registrierten" Denial-of-Service-Angriff (DDoS) mit [...] 7,3 Terabit pro Sekunde (TBit/s) blockiert. [...] Diese Attacke war demnach rund 12 Prozent größer als der vorherige Rekord und lieferte ein massives Datenvolumen von 37,4 Terabyte in nur 45 Sekunden. [...]*

*Stellen Sie sich vor, Sie könnten mit Ihrem Smartphone 12,5 Millionen hochauflösende Fotos schießen und hätten nie einen vollen Speicherplatz." Und das alles in 45 Sekunden.*

*[...] Mitgewirkt hätten über 122.145 Quell-IP-Adressen, die sich über 5433 autonome Netzwerksysteme in 161 Ländern erstreckten.*

*—22.06.2025 Heise.de - Rekord DDoS Angriff*

# Distributed Denial-of-Service Angriffe - Beispiele

- **TCP Stack Attacks** SYN, FIN, RST, ACK, SYN-ACK, URG-PSH, other combinations of TCP Flags, slow TCP attacks
- **Application Attacks:** HTTP GET/POST Floods, slow HTTP Attacks, SIP Invite Floods, DNS Attacks, HTTPS Protocol Attacks
- **SSL/TLS Attacks:** Malformed SSL Floods, SSL Renegotiation, SSL Session Floods
- **DNS Cache Poisoning**
- **Reflection Amplification Flood Attacks:** TCP, UDP, ICMP, DNS, mDNS, SSDP, NTP, NetBIOS, RIPv1, rpcbind, SNMP, SQL RS, Chargen, L2TP, Microsoft SQL Resolution Service
- **Fragmentation Attacks:** Teardrop, Targa3, Jolt2, Nestea
- **Vulnerability Attacks**
- **Resource Exhaustion Attacks:** Slowloris, Pyloris, LOIC, etc.
- **Flash Crowd Protection**
- **Attacks on Gaming Protocols**

# Schutz vor DDoS-Angriffen: On-Site Maßnahmen

- Aufrüsten der Ressourcen (z. B. Bandbreite, CPU, RAM, ...)
- Exemplarische Sofortmaßnahmen bei aktivem Angriff:
  - Whitelisting von IP-Adressen von besonders wichtigen Clients
  - Blacklisting von IP-Adressen aus bestimmten Bereichen
  - Captchas
  - Überprüfung der Browser-Echtheit
- Anti-DDos Appliances

## Achtung!

Diese Maßnahmen sind häufig teuer und ggf. begrenzt effektiv; wenn der Angriff die verfügbare Bandbreite übersteigt, sind diese Maßnahmen darüber hinaus wirkungslos.

# Schutz vor DDoS-Angriffen: Off-Site Maßnahmen

- Einbinden des ISP
- Einbinden spezialisierter Dienstleister  
(Im Angriffsfall wird mittels BGP-Rerouting der Traffic an den Dienstleister umgeleitet, der dann die DDos Attacke filtert.)
- Content-Delivery-Networks (CDNs) für statische Inhalte (z. B. Cloudflare, Akamai, ...)
- Distributed Clouds

# Password Sniffing

In der Anfangszeit: unverschlüsselte Übertragung von Passwörtern (telnet, ftp, ...)

In der Übergangszeit:

Verwendung von Einmal-Passwörtern (S/Key, ...)

Heute:

Passwörter werden verschlüsselt übertragen (ssh, https, ...)

Zusätzliche Absicherung durch Zwei-Faktor-Authentifizierung  
(basierend auf Einmalpassworten: TOTP, ...)

---

Unverschlüsselte Passworte können leicht mittels eines Sniffers, der den Netzwerkverkehr mitschneidet (z. B. Wireshark), abgefangen werden.

# Einmal-Passwörter

Die Idee ist, dass Passwörter nur genau einmal gültig sind und nicht wiederverwendbar sind.

- Tokens (z. B. RSA SecurID)
- Codebuch: Liste von Einmal-Passwörtern, die das gemeinsame Geheimnis sind.
- S/Key: Passwort „wird mit einem Zähler kombiniert“ und dann gehasht.

# Das S/Key Verfahren

Einmal-Passwort-System nach Codebuch-Verfahren.

## Initialisierung

1. Der Nutzer gibt sein Passwort  $W$  ein; dies ist der geheime Schlüssel.  
(Sollte  $W$  bekannt werden, dann ist die Sicherheit des Verfahrens nicht mehr gewährleistet.)
2. Eine kryptografische Hash-Funktion  $H$  wird  $n$ -mal auf  $W$  angewandt, wodurch eine Hash-Kette von  $n$  einmaligen Passwörtern entsteht.  $H(W), H(H(W)), \dots, H^n(W)$
3. Das initiale Passwort wird verworfen.
4. Der Benutzer erhält die  $n$  Passwörter, die in umgekehrter Reihenfolge ausgedruckt werden:  $H^n(W), H^{n-1}(W), \dots, H(H(W)), H(W)$ .
5. Nur das Passwort  $H^n(W)$ , das an erster Stelle der Liste des Benutzers steht, der Wert von  $n$  und ggf. ein Salt, wird auf dem Server gespeichert.

## Anmeldung

Identifiziere das letzte verwendete Passwort  $n$ .

- Der Server fragt den Nutzer nach dem Passwort  $n - 1$  (d. h.  $H^{n-1}(W)$ ) und übermittelt ggf. auch den Salt.
- Der Server hasht das Passwort und vergleicht es dann mit dem gespeicherten Passwort  $H^n(W)$ .
- Ist das Passwort korrekt, dann wird der Nutzer angemeldet und der Server speichert das Passwort  $H^{n-1}(W)$  als neues Passwort  $H^n(W)$  und dekrementiert  $n$ .

---

Im Original basiert S/Key auf der kryptographischen Hashfunktion MD4. Ein Austausch wäre aber selbstverständlich möglich!

Intern verwendet S/KEY 64-bit Zahlen. Für die Benutzbarkeit werden diese Zahlen auf sechs kurze Wörter, von ein bis vier Zeichen, aus einem öffentlich zugänglichen 2048-Wörter-Wörterbuch ( $2048 = 2^{11}$ ) abgebildet. Zum Beispiel wird eine 64-Bit-Zahl auf "ROY HURT SKI FAIL GRIM KNEE" abgebildet.

## HMAC-based one-time password (HOTP)[11]

- ermöglicht die Erzeugung von Einmal-Passwörtern auf Basis eines geheimen Schlüssels und eines Zählers; Parameter:
  - Ein kryptografisches Hash-Verfahren  $H$  (Standard ist SHA-1)
  - einen geheimen Schlüssel  $K$ , der eine beliebige Bytefolge ist
  - Ein Zähler  $C$ , der die Anzahl der Iterationen zählt
  - Länge des Passworts:  $d$  (6-10, Standardwert ist 6, empfohlen werden 6-8)
- Zur Authentifizierung berechnen beide das Einmalpasswort (HOTP) und dann vergleicht der Server den Wert mit dem vom Client übermittelten Wert:

Berechnung aus dem Schlüssel  $K$  und dem Zähler  $C$ :

$$HOTP(K, C) = truncate(HMAC_H(K, C))$$

$$truncate(MAC) = extract31(MAC, MAC[(19 \times 8 + 4) : (19 \times 8 + 7)])$$

$$HOTP\ value = HOTP(K, C) \bmod 10^d \quad (\text{führende Nullen werden nicht abgeschnitten})$$

---

*truncate* verwendet die 4 niederwertigsten Bits des MAC als Byte-Offset  $i$  in den MAC. Der Wert 19 kommt daher, dass ein SHA-1 160 Bit hat und  $160/8 = 20$  Byte.

*extract31* extrahiert 31 Bit aus dem MAC. Das höchstwertig Bit wird (wenn es nicht 0 ist) entsprechend maskiert. Eine Schwäche des Algorithmus ist, dass beide Seiten den Zähler erhöhen müssen und, falls die Zähler aus dem Tritt geraten, ggf. eine Resynchronisation notwendig ist.

---

[12]<https://www.rfc-editor.org/rfc/rfc4226>



## Time-based one-time password (TOTP)[12]

- Erzeugung von zeitlich limitierten Einmal-Passwörtern (z. B. 30 Sekunden)
- Basierend auf einem vorher ausgetauschten geheimen Schlüssel und der aktuellen Zeit  
Z. B. Unix-Zeit in Sekunden (ganzzahlig) und danach gerundet auf 30 Sekunden.
- Es wird das HOTP Verfahren mit der Zeit als Zähler verwendet und entweder SHA-256 oder SHA-512 als Hashverfahren, d. h.  $\text{TOTP value}(K) = \text{HOTP value}(K, C_T)$ , wobei  $T$  die „aktuelle Zeit“ ist.

$$C_T = \lfloor \frac{T - T_0}{T_X} \rfloor$$

$T_X$  ist die Länge eines Zeitintervalls (z. B. 30 Sekunden)

$T$  ist die aktuelle Zeit in Sekunden seit einer bestimmten Epoche

$T_0$  ist bei Verwendung der Unix-Zeit 0

$C_T$  ist somit die Anzahl der Dauern  $T_X$  zwischen  $T_0$  und  $T$

---

Das Verfahren verlangt somit, dass die Uhren von Server und Client (hinreichend) synchronisiert sind.

---

[13]<https://www.rfc-editor.org/rfc/rfc6238>

# Secure Shell (SSH)

## Verschlüsselte Verbindung

SSH ermöglicht die sichere Fernanmeldung von einem Computer bei einem anderen (typischerweise über TCP über Port 22). Es bietet mehrere Optionen für eine starke Authentifizierung und schützt die Sicherheit und Integrität der Kommunikation durch starke Verschlüsselung

## Ablauf

1. Authentisierung des Server-Rechners
2. Authentisierung des Benutzers (bzw. des Clients) mittels
  - a. Passwort
  - b. ~~hosts~~-Eintrag
  - c. privatem (RSA-)Key (hauptsächlich verwendete Methode)
3. Kommunikation über symmetrisch verschlüsselte Verbindung

---

Die Authentifizierung mittels eines Schlüsselpaars dient primär der Automatisierung (dann wird auch keine „Schlüsselphrase“ zum Schutz des Passworts verwendet). Auf jeden Fall ist effektives Schlüsselmanagement erforderlich:

*[...] In einigen Fällen haben wir mehrere Millionen SSH-Schlüssel gefunden, die den Zugang zu Produktionsservern in Kundenumgebungen autorisieren, wobei 90 % der Schlüssel tatsächlich ungenutzt sind und für einen Zugang stehen, der zwar bereitgestellt, aber nie gekündigt wurde.*

—SSH.com (Dez. 2023)

# Secure Shell (SSH) - Protokoll

Beide Seiten haben einen Public-private Key Schlüsselpaar zur gegenseitigen Authentifizierung

**User Keys:** ■ `Authorized keys` - Serverseitige Datei mit den öffentlichen Schlüsseln der Nutzer

■ `Identity keys` - private Schlüssel der Nutzer

**Host keys:** dienen der Authentifizierung von Servern (verhindern Person-in-the-Middle-Angriffe)

**Session Keys:** werden für die symmetrische Verschlüsselung der Daten in einer Verbindung verwendet. Session Keys (🇩🇪 *Sitzungsschlüssel*) werden während des Verbindungsaufbaus ausgehandelt.

---

Im Falle von SSH gibt es kein initiales Vertrauen zwischen Server und Client.

# Secure Shell (SSH) - Verbindungsaufbau - Beispiel

```
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to example.org [1.2.3.4] port 22.
debug1: Connection established.
debug1: identity file /home/user/.ssh/id_rsa type -1
debug1: identity file /home/user/.ssh/id_rsa-cert type -1
debug1: identity file /home/user/.ssh/id_dsa type -1
debug1: identity file /home/user/.ssh/id_dsa-cert type -1
debug1: Remote protocol version 1.99, remote software version OpenSSH_5.8
debug1: match: OpenSSH_5.8 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.5p1 Debian-6
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server→client aes128-ctr hmac-md5 none
debug1: kex: client→server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host 'example.org' is known and matches the RSA host key.
debug1: Found key in /home/user/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: Roaming not allowed by server
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password,keyboard-interactive,hostbased
debug1: Next authentication method: publickey
debug1: Trying private key: /home/user/.ssh/id_rsa
debug1: Trying private key: /home/user/.ssh/id_dsa
debug1: Next authentication method: keyboard-interactive
debug1: Authentications that can continue: publickey,password,keyboard-interactive,hostbased
debug1: Next authentication method: password
user@example.org's password:
debug1: Authentication succeeded (password).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = en_US.UTF-8
```

# Secure Shell (SSH) - Risiken durch mangelnde Schlüsselverwaltung

- Schlüssel werden nicht regelmäßig ausgetauscht
- Schlüssel werden nicht gelöscht, wenn sie nicht mehr benötigt werden
- viele (die meisten) Schlüssel werden nicht verwendet
- Es ist oft nicht bekannt, wer Zugriff auf welche Schlüssel hat(te)
- Es ist nicht bekannt, welche Schlüssel auf welche Systeme Zugriff haben
- Malware kann SSH-Schlüssel stehlen
- SSH Keys können ggf. privilegierten Zugriff gewähren
- SSH Keys können benutzt werden, wenn um Backdoors zu verstecken
- Server keys erlauben ggf. Person-in-the-Middle-Angriffe

# Schwachstellen in SSH

## ***Nearly 11 million SSH servers vulnerable to new Terrapin attacks***

*[...] It [The Terrapin attack] manipulates sequence numbers during the handshake process to compromise the integrity of the SSH channel, particularly when specific encryption modes like ChaCha20-Poly1305 or CBC with Encrypt-then-MAC are used. [...]*

*By Bill Toulas*

*—January 3, 2024 10:06 AM*

# Übung

## 6.1. Port Scans - IDLE Scan

- Warum kann bei einem IDLE Scan nicht festgestellt werden weshalb ein Port geschlossen oder gefiltert ist?
- Welchen Wert hat die IP ID des Zombies, der einem IDLE Scan durchführt, wenn der Zielpport offen bzw. geschlossen ist, wenn der Scanner diesen wieder abfragt?

# Übung

## 6.2. S/Key

1. Welche Vorteile bieten Einmalpasswortsysteme gegenüber Systemen mit mehrfach zu verwendenden Passwörtern?
2. Welchen Angriffen sind Einmalpasswortsysteme weiterhin ausgesetzt?
3. Wenn ein Passwort  $H^L(W)$ ,  $1 < L < N$  bekannt ist, welche Auswirkungen hat dies auf die Sicherheit des Verfahrens?



# Übung

## 6.3. TOTP

Identifizieren Sie die Vor- und Nachteile von TOTP gegenüber S/Key und fragen Sie sich an welcher Stelle es (aus Sicherheitsperspektive) mögliche Schwächen gibt?

Die Standardzeitspanne ist 30 Sekunden. Welche Konsequenzen hätte eine deutliche Verlängerung bzw. Verkürzung der Zeitspanne?

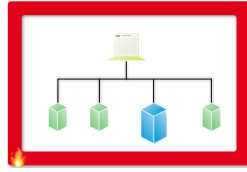
# Übung

## 6.4. DDoS

1. Welches Problem entsteht wenn zum Schutze vor Angriffen auf die Verfügbarkeit die Ressourcen von IT-Systemen und deren Internet-Anbindung erhöht werden?
2. Recherchieren Sie was ein „Low and Slow Angriff“ ist.



# Unabhängiges Netz - „Ideale Situation“



**Vorteile:** ■ keinerlei Angriffsmöglichkeiten von außen

**Nachteile:** ■ kein Schutz gegen Insider  
■ kein Zugang zum Internet

---

Wie bereits diskutiert gibt es auch Angriffsmuster gegen Air-Gapped-Systeme. Ein Beispiel ist der Stuxnet-Wurm, der sich initial über USB-Sticks verbreitet.

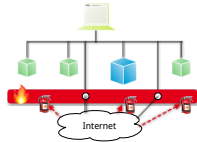
Wenn man kein Zugang zum Internet hat, dann hat man zum Beispiel kein Zugriff auf externe Dienste wie NTP und das Einspielen von Updates ist nur über Umwege möglich.

# Von der Notwendigkeit des Schutzes von Rechnern

*[...] Züger und sein Team hätten [...] erst kürzlich ein Experiment durchgeführt, [...]. Sie hätten einen Computer "ohne jeglichen Schutz" mit dem Internet verbunden, um zu sehen, wie lange es dauere, bis er befallen sei. Konkrete Details zur Konfiguration dieses Systems werden zwar nicht genannt, angeblich war der Rechner aber schon nach 20 Minuten infiltriert.*

—Golem.de 6.2.2024

# Schutzschicht zwischen internem und externem Netz

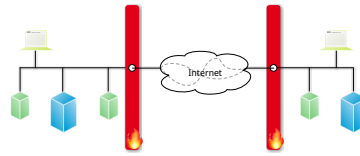


- Kontrolle des Nachrichtenverkehrs durch Filterung
- begrenzte Isolation mit begrenztem Schutz

---

Eine Firewall schafft zwischen verbundenen Netzen Sicherheitsdomänen mit unterschiedlichem Schutzbedarf. Eine wichtige Teilaufgabe ist das Ausarbeiten von Sicherheitsrichtlinien.

# Realisierung von Virtual Private Networks (VPN)



- Aufbau einer scheinbar privaten Verbindung von Firmennetzen über das (öffentliche) Internet.
- Zusätzliche Verbindungsverschlüsselung zwischen den Firewalls.

-----

Ziel ist es aktive und passive Angriffe zu unterbinden. Selbst bei verschlüsselten Verbindungen kann die Verkehrsflussanalyse noch Informationen liefern über die Verbindungen liefern.

# Kommerzielle VPNs für Endnutzer



---

## Motivation

- Schutz der Privatsphäre; der ISP kennt nicht mehr die Webseiten, die man aufruft.
- Die IP-Adresse des Nutzers ist den aufgerufenen Webseiten nicht mehr bekannt und kann deswegen der Umgehung von Geo-Blocking dienen.

## Nachteile?

- Vertrauen in den VPN-Anbieter muss vorhanden sein. Insbesondere, beim Einsatz zum Stärken der Privatsphäre, muss der VPN-Anbieter vertrauenswürdig sein und sollte ein so genannter „no-log“ Anbieter sein.
- Es gibt auch (scheinbar kostenlose) VPN-Anbieter, die die Daten der Nutzer dann aber verkaufen (ehemals: **Facebook Onavo**).



# Schutz auf den Schichten des TCP/IP Stacks

Zentraler Schutz des gesamten internen Netzwerks durch:

- Paket Filter (🚧 *Packet Filtering*)
  - Blockieren bestimmter IP-Empfänger-Adressen (extern / intern)
  - Blockieren bestimmter IP-Absender-Adressen (extern / intern)  
(z. B. aus dem Internet mit internen IP-Absender-Adressen)
  - Blockieren bestimmter Dienste; ggf. nur für bestimmte IP-Adressen
- Filter auf Anwendungsebene (🚧 *Application-level Filtering*)
  - inhaltsbezogene Filterung der Verkehrsdaten eines Dienstes  
(z. B. Virenfilter oder Spamfilter)
  - wirkungslos bei verschlüsselten Verkehrsdaten
- Protokollierungsmöglichkeit der Kommunikation von / nach extern

---

Firewalls (alleine) können die Struktur des Netzwerks nicht verbergen.

# DoS Attacke auf Anwendungsebene

## *[...] Angriff auf die Kleinen*


*Waren bei früheren Spamangriffen massenhaft Accounts auf der größten Mastodon-Instanz `mastodon.social` angelegt worden, die dann von dort ihre Inhalte verbreiteten, trifft es nun nicht die größte, sondern die kleinsten. Automatisiert werden dabei Instanzen ausgesucht, auf denen eine Registrierung ohne Überprüfung und sogar ohne ein Captcha möglich ist. Das können etwa solche mit wenigen Accounts sein, die von Enthusiasten etwa für eine Gemeinde betrieben werden. Waren die Verantwortlichen in den vergangenen Tagen nicht aufmerksam, wurden diese Instanzen dann regelrecht überrannt. Die Spam-Accounts verschickten massenhaft Nachrichten mit einem Bild des namensgebenden Frühstücksfleischs und Links zu Discord-Servern, die wohl lahmgelegt werden sollten.*

*—Mastodon: Spamwelle zeigt Schwächen auf [...]*

# Realisierungsmöglichkeiten von Firewalls

- Hardware-Firewall
  - Screening Router
  - Application Gateway (auch Bastion Host)
    - Proxy-Server für bestimmte Dienste
    - Server-Software
    - Client-Software (HTTP-Browser, telnet, ftp, ...) kommuniziert nur mit dem Application Gateway; erscheint gegenüber Client Software als Endpunkt.
- Software-Firewall (*Personal Firewall*)

---

Im Falle eines  *Bastion Host*, ist dies der einzige unmittelbar aus dem Internet erreichbare Rechner.

# Dual-Homed Host

## Aufbau

- zwei Netzwerkkarten: ggf. private interne Adressen
- Screening Router & Gate: Packet Filter und Application-Level Filter
- Proxy-Dienste installieren
- Benutzer-Logins von extern



### !! Wichtig

Bei der Konfiguration der Netzwerkkarten gilt:

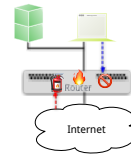
*IP-Pakete nicht automat. weiterleiten*

# Screening Router

## Aufbau

Programmierbarer Hardwarerouter mit  
einfachen Filterfunktionen:

- nur Paket-Header prüfen
- schnelle Auswertung ermöglicht hohen Durchsatz
- Realisierung eines Packet Filters



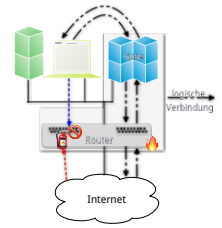
## Bewertung

✓ einfach und billig	! schwer zu testen
✓ flexibel	! Protokollierung
	! Fernwartung
	! keine Inhaltsfilterung

# Screened Host

## Aufbau

- Screening Router blockiert:
  - Pakete von / an interne Rechner (nicht Gate)
  - Source-Routed Pakete
- von extern nur Gate sichtbar
- Pakete von intern nur via Gate
- Gate bietet Proxy-Server (z. B. für E-Mail)



---

*Source-Routed Pakete* sind Pakete, die den Weg durch das Netzwerk explizit angeben. (*Source-routing* wird auch als *Path Addressing* bezeichnet und wird im Allgemeinen als Sicherheitsproblem angesehen.)

Gibt es für eine bestimmte Anwendung kein Application-level Proxy, dann kann auf einen für TCP/UDP generischen Proxy zurückgegriffen werden. Dieser arbeitet auf dem Session Layer und kann nur die Header-Informationen auswerten. Es handelt sich dann um ein *Circuit-level Proxy/Gateway*. Im Vergleich zu einem Application-level Proxy ist die Sicherheit geringer, da der Circuit-level Proxy nicht in der Lage ist, die Daten zu interpretieren.

Ein allgemeines Problem ist, dass viele Anwendungen auf generische Protokolle wie HTTP aufsetzen. Weiterhin betreiben einige Anwendungen „Port Hopping“, d. h. sie wechseln den Port wenn der Standardport nicht offen ist.

Eine Anforderung an „Next-generation Firewalls“ ist, dass diese die Analyse von den Daten einer Anwendung unabhängig vom Port und Protokoll ermöglichen.

# Konfiguration eines Gateways

Das Ziel der Konfiguration muss eine minimale angreifbare Oberfläche sein.

- Abschalten aller nicht-benötigten Netzdienste
- Löschen aller nicht benötigter Programme
- Rechte von `/bin/sh` auf 500 setzen
- Rechte aller Systemverzeichnisse auf 711 setzen
- keine regulären Benutzerkennungen
- root-Login mit Einmal-Passwortsystem bzw. 2-Faktor Authentifizierung
- setzen von Platten- und Prozess-Quotas
- volle Protokollierung, möglichst auf Hardcopy-Gerät
- möglichst sichere, stabile und regelmäßig aktualisierte Betriebssystemversion einsetzen

---

Die Rechte von `/bin/sh` auf 500 setzen bedeutet, dass nur der Eigentümer (root) es ausführen kann.

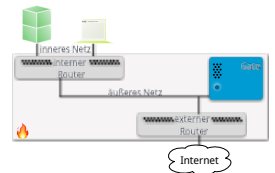
Default:

```
$ ls -al /bin/sh
-rwxr-xr-x 1 root wheel 101232 Oct 1 06:10 /bin/sh
```

# Screened Subnet

## Aufbau

- interner Screening Router als weiterer Schutzwall
  - blockiert Dienste, die nicht einmal bis zum Gate gelangen sollen
  - lässt nur Pakete zum / vom Gate durch
- äußeres Netz realisiert Demilitarisierte Zone (DMZ) für HTTP-Server, Mail-Server, ...





# Intrusion Detection Systeme (IDS)

## Definition

Ein IDS ist ein Gerät (meist ein speziell konfigurierter Rechner), das vielfältige Techniken zur Erkennung von Angriffen anwendet und Angriffe meldet und ggf. abwehrt, in dem (z. B.) die Firewall automatisch umkonfiguriert wird.

## Motivation

- Firewalls alleine sind zu statisch und deswegen häufig nicht ausreichend
- bessere Aufzeichnung und flexiblere Erkennung notwendig
- angepasste Reaktion notwendig

## Umsetzung

An verschiedenen, neuralgischen Stellen werden spezielle Sensoren platziert, die (hier) den Netzwerkverkehr überwachen und verdächtige Aktivitäten melden.

---

Miteinander verwandt bzw. typischerweise in einem Produkt zu finden:

- Intrusion Detection (IDS)
- Intrusion Response (IRS)
- Intrusion Prevention (IPS)


# IDS-Erkennungstechniken

- Signaturerkennung
- statistische Analyse
- Anomalieerkennung

## Probleme

- Fälschlicherweise gemeldete Angriffe (false positives)
- nicht gemeldete Angriffe (false negatives) (insbesondere bei neuartigen Angriffen)
- Echtzeitanforderung, insbesondere bei Hochgeschwindigkeitsnetzen
- Aufzeichnung bei Netzwerken mit Switches ( ⇒ spez. SPAN Port)
- Sensoren sollen unbeobachtbar sein (*stealth*)

---

SPAN ( *Switched Port Analyzer*) Ports sind spezielle Ports auf Switches, die bestimmten Verkehr (z. B. bestimmte Pakete) die über ein Switch gehen, an einen definierten Port weiterleiten können. An diesem Port kann dann eine Analyse des Verkehrs durchgeführt werden / ein Sensor angeschlossen werden.

# Übung

## 6.5. Firewalls

1. Was sind Vorteile eines Dual Homed Host gegenüber einem Paketfilter? Was sind die Nachteile?
2. Benennen Sie zwei konzeptionelle Grenzen von Firewalls. D. h. zwei Szenarien gegen die Firewalls nicht schützen können.
3. Für welche der folgenden Cybersicherheitsstrategien können Firewalls eingesetzt werden:
  1. Angriffe vermeiden
  2. Angriffe erkennen
  3. Angriffe abwehren/Angriffen entgegenwirken
  4. Reaktion auf Angriffe
4. Sie werden beauftragt die Firewall so einzurichten, dass Mails mit Schadsoftware nicht durchgelassen werden. Wie reagieren Sie?