Kryptografie

Prof. Dr. Henning Pagnia

DHBW Mannheim

Frühjahr 2022

Wichtiges zur Vorlesung

Prof. Dr. Henning Pagnia

- Wirtschaftsinformatik
- Email: *pagnia@dhbw-mannheim.de*
- Telefon: *0621 / 4105-1131*
- Raum: *149 B*

Begriffe

Begriffsklärung

- Kryptografie
- Kryptoanalyse ⇒ Kryptologie als Sammelbegriff
- codieren ⇒ Fehlererkennung bzw. -korrektur: z. B. CRC
- verschlüsseln ⇒ Geheimhaltung: z. B. AES

Einordnung

- Datensicherheit:
  Schutz digitaler und analoger Daten
  ⇒ Informationssicherheit
  Best Practices z. B. BSI-Grundschutz
- Datenschutz:
  Schutz der personenbezogenen Daten eines Menschen
  ⇒ informationelle Selbstbestimmung
  Regelungen u. a. in der EU-DSGVO und dem BDSG

Übersicht

Themengebiete der Vorlesung

- Einfache Chiffren
- Modulare Arithmetik
- Symmetrische Verschlüsselung
- Schlüsseltauschverfahren
- Asymmetrische Verschlüsselung
- Kryptografische Hashverfahren
- Digitale Signaturen
- Authentisierungsprotokolle

Literatur

[Beut2015] *Albrecht Beutelspacher*: Kryptologie – Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen, 10. Auflage.
Springer Spektrum, 2015. ISBN 978-3-658-05975-0.

[Buch2016] *Johannes Buchmann*: Einführung in die Kryptographie, 6. Auflage.
Springer Spektrum, 2016. ISBN 978-3-642-39774-5.

[Ecke2018] *Claudia Eckert*: IT-Sicherheit: Konzepte – Verfahren – Protokolle, 10. Auflage.
De Gruyter Oldenbourg, 2018. ISBN 978-3-11-055158-7.

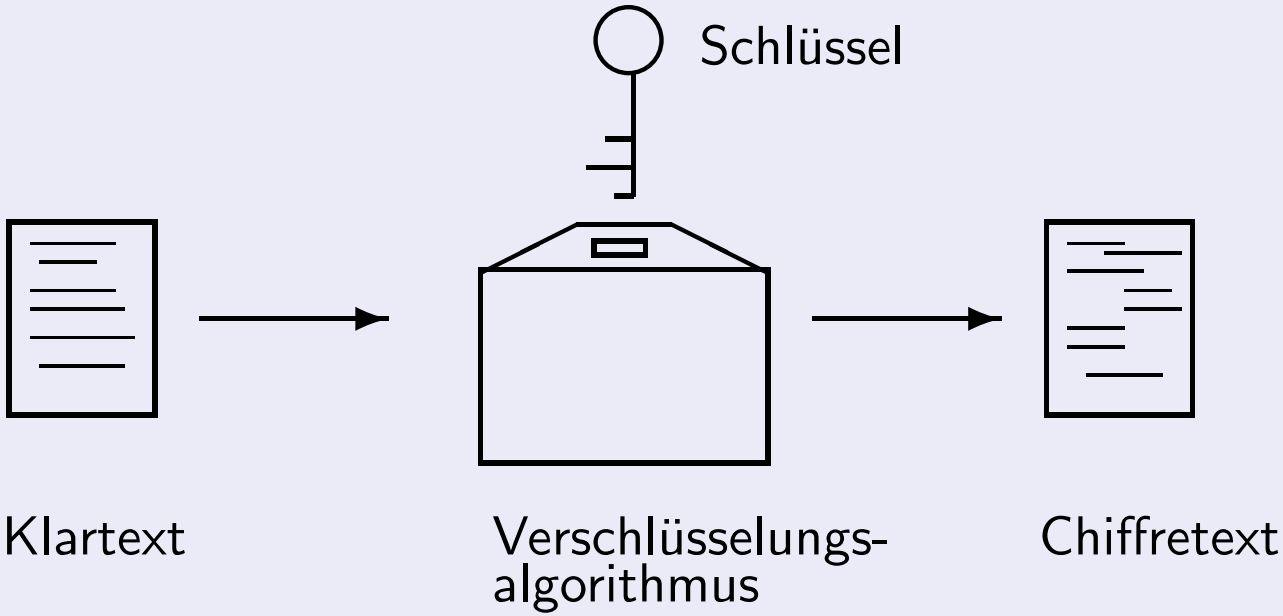
[PP2016] *Christof Paar und Jan Pelzl*: Kryptografie verständlich – Ein Lehrbuch für Studierende und Anwender.
Springer Vieweg, 2016. ISBN 978-3-662-49296-3.

[Schm2016] *Klaus Schmeh*: Kryptografie, 6. akt. Auflage.
dpunkt.verlag, 2016. ISBN-13: 978-386490-356-4.

[Schn1995] *Bruce Schneier*: Applied Cryptography, 2nd edition.
John Wiley & Sons, 1995. ISBN-13: 978-0-471-11709-4.

Verschlüsselung

Vorgehensweise



Bezeichner

- Klartext: $\boldsymbol{m}$ (message, plain text)
- Schlüssel: $\boldsymbol{k}$ (key)
- Chiffretext: $\boldsymbol{c}$ (cipher text)

Verschlüsselung (Forts.)

Schutzziele der Verschlüsselung

- Privacy (Geheimhaltung)
- Authenticity (Authentizität)
- Integrity (Integrität)
- Non-Repudiation (Nichtabstreitbarkeit)
- Perfect Forward Secrecy (in etwa: zukünftige Geheimhaltung)
  - *der Verlust eines einzigen Schlüssels führt nicht zum Verlust der Vertraulichkeit der gesamten Kommunikation*

Kryptografie

Anwendungsgebiete

- sichere lokale Speicherung
  - Verschlüsselung von einzelnen Dateien
  - Verschlüsselung ganzer Dateisysteme
- sichere Nachrichtenübertragung über einen unsicheren Nachrichtenkanal

Historisch

- Das traditionelle Einsatzgebiet von Verschlüsselung ist das Militärische
- Chiffrierung und Dechiffrierung wurden ohne den Einsatz von Rechenmaschinen zumeist manuell durchgeführt
- Es wurden folgende Vereinbarungen getroffen, um das Brechen der Chiffren zu erschweren:
  - Auf Satzzeichen und Leerzeichen wird verzichtet: Worte werden direkt hintereinander geschrieben
  - Auf Groß-/Kleinschreibung wird verzichtet: Klartexte werden in Großbuchstaben angegeben, Chiffretexte in Kleinbuchstaben
  - moderne Chiffren nutzen derartige Einschränkungen i. Allg. nicht

Kryptografie (Forts.)

Beispiel

Versuchen Sie den folgenden Chiffretext zu entschlüsseln:

```
idaenablqudnbbnudwpenafnwmnwrlqcmnwlnbja
```

Kryptografie (Forts.)

Einige mögliche Angriffe

- Brute Force
  durchprobieren aller möglichen Schlüssel
- Cipher Text Only
  nur der Chiffretext einer Nachricht liegt vor (Angriff i. Allg. sehr schwierig!)
  - liegen mehrere Chiffetexte vor, die mit dem identischen Schlüssel verschlüsselt wurden, führt das Finden des Schlüssels zur Entschlüsselung aller Chiffretexte
- Known Plain Text
  errechnen des Schlüssels, wenn Chiffretext und Klartextteile bekannt sind
- Chosen Plain Text
  bei Zugriffsmöglichkeit auf die Verschlüsselungsmaschine generiert der Angreifer eigene Klartexte (z. B. Folgen von Null-Bits) und erhält die dazugehörigen Chiffretexte

Kryptografie (Forts.)

Klartext des Beipiels

ZURVERSCHLUESSELUNGVERWENDENICHTDENCAESAR

Es handelt sich um eine sehr schwache Verschlüsselung: eine Verschiebe-Chiffre. Dabei werden die Klartextzeichen im Alphabet um jeweils k Buchstaben verschoben.
Mittels Brute Force oder durch eine Frequenzanalyse lässt sich die Verschlüsselung brechen:
Der häufigste Buchstabe ist hier das ***N*** (8-mal). Wenn man davon ausgeht, dass der Text ein deutscher Text ist, dann ist es sehr wahrscheinlich, dass das ***E*** in ein ***N*** überführt wurde. ***E*** ist nämlich der häufigste Buchstabe in der deutschen Sprache. Zum Überprüfen dieser Hypothese versucht man also (erfolgreich) eine Verschiebung des Klartextalphabets um $k = 9$ Buchstaben.

Wichtig: das Kerckhoffs Prinzip (1883)

Die Sicherheit der Verschlüsselung darf nur von der Geheimhaltung des Schlüssels abhängen – nicht von der Geheimhaltung des Algorithmus!

Kryptografie (Forts.)

Formale Definition: *Kryptosystem*

Ein *Kryptosystem* $\boldsymbol{KS}$ ist ein Fünf-Tupel

$$\boldsymbol{KS} = (\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{K}, \boldsymbol{E}, \boldsymbol{D})$$

mit

- $\boldsymbol{M}$ ist eine endliche Menge, das Klartextalphabet
- $\boldsymbol{C}$ ist eine endliche Menge, das Chiffretextalphabet
- $\boldsymbol{K}$ ist eine endliche Menge, der Schlüsselraum
- $\boldsymbol{E}$ ist eine endliche Menge, die für jedes $\boldsymbol{k} \in \boldsymbol{K}$ eine Verschlüsselungsfunktion $\boldsymbol{e_k} : \boldsymbol{M} \rightarrow \boldsymbol{C}$ definiert
- $\boldsymbol{D}$ ist eine endliche Menge, die für jedes $\boldsymbol{k} \in \boldsymbol{K}$ eine Entschlüsselungsfunktion $\boldsymbol{d_k} : \boldsymbol{C} \rightarrow \boldsymbol{M}$ definiert
- dabei gilt:

$$\boldsymbol{d_k}(\boldsymbol{e_k}(\boldsymbol{m})) = \boldsymbol{m} \quad \forall \boldsymbol{m} \in \boldsymbol{M}$$

Kryptografie (Forts.)

Wie sicher ist ein Kryptosystem?

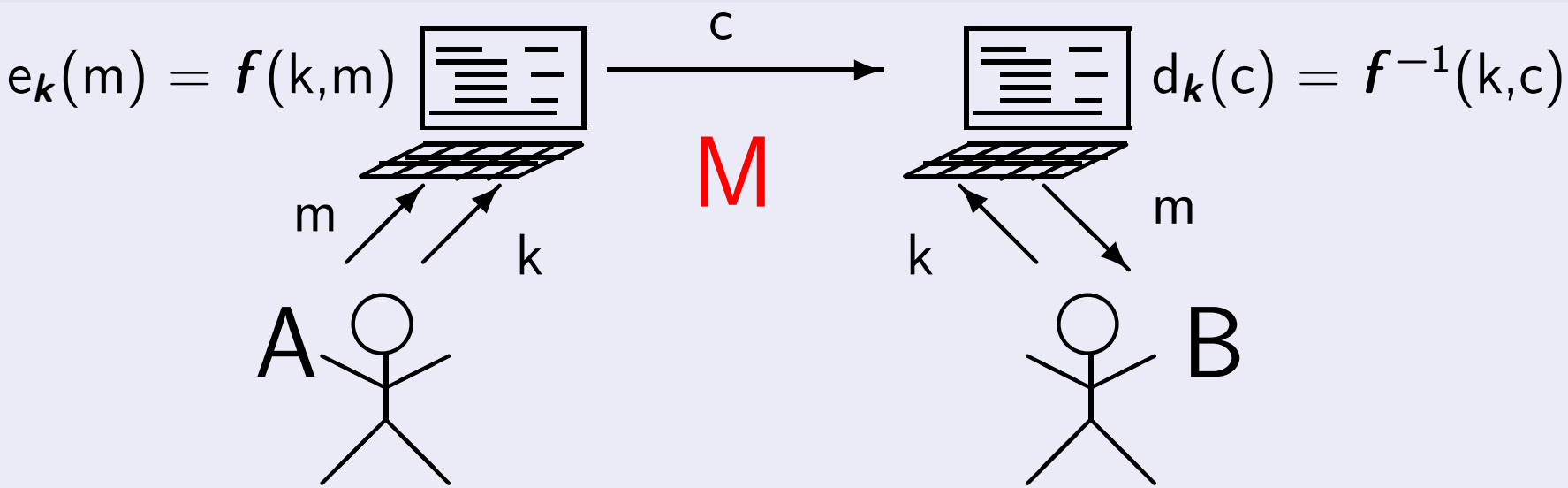
- *perfekt sicher*
  ein Kryptosystem, das jeder kryptoanalytischen Attacke widersteht – auch wenn dem Angreifer eine uneingeschränkte Rechenleistung zur Verfügung steht
- *berechnungssicher* (computationally secure)
  ein Kryptosystem, das mit einer kryptoanalytischen Attacke bei begrenzter Rechenleistung in der Praxis nicht in einer *erlebbaren Zeit* gebrochen werden kann
- *unsicher*
  alle anderen Kryptosysteme

Klassifizierung der Verfahren

- symmetrisch / asymmetrisch
  $\Rightarrow$ Kombination: hybrid
- Blockchiffre / Stromchiffre

Symmetrische Verschlüsselung

Prinzip



- Absender: ***A*** (Alice)
- Empfänger: ***B*** (Bob)
- Angreifer: ***M*** (Mallory)

Unzählige Verfahren

- IDEA, DES, Skipjack, RC4, RC5, Blowfish, RC6,Threefish, **AES**, ...

Verschiebe-Chiffren

Definition

- Seien $m, c, k \in \{0, 1, \ldots, 25\}$:

  Verschlüsselung: $e_k(m) \equiv (m + k) \bmod 26$

  Entschlüsselung: $d_k(c) \equiv (c - k) \bmod 26$

- mit der folgenden Codierung für die Buchstaben des Alphabets:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Anmerkungen

- es handelt sich um eine symmetrische Block-Chiffre mit der Blocklänge 1
- $\equiv$ ist die Kongruenzrelation
- die Menge $\{0,1,\ldots,25\}$ bezeichnet man als $\mathbb{Z}_{26}$
- im Allgemeinen setzt man $k = 3$ für den historischen Caesar-Chiffre

Substitution-Chiffren

Vorgehensweise

- Für den Chiffretext wird jeder Klartextbuchstabe wird jeweils immer durch einen bestimmten anderen ersetzt
- Als Schlüssel dient eine Tabelle: In der oberen Zeile stehen die Buchstaben des Klartextalphabets, in der unteren eine Permutation derselben
- Beispiel:

A	B	C	D	E	F	G	H	I	J	K	L	M
u	g	y	b	l	m	r	q	h	w	z	a	j

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
c	i	k	x	o	e	s	t	f	v	p	n	d

  Wie lautet der Klartext zu

  s i k e l y o l s

- Verschiebe-Chiffren sind Spezialfälle von Substitution-Chiffren
- Brechen einer einfachen Substitution-Chiffre ist i. Allg. mittels einer Frequenzanalyse möglich

Permutation-Chiffren

Vorgehensweise

- Für den Chiffretext wird die Buchstabenfolge des Klartextes permutiert
- Beispiel:

koeee geeam csrle ihiwn fvhsy oinrn istel epgnf udscu rvlti eunis hosen

Permutation-Chiffren

Vorgehensweise

- Für den Chiffretext wird die Buchstabenfolge des Klartextes permutiert
- Beispiel:

koeee geeam csrle ihiwn fvhsy oinrn istel epgnf udscu rvlti eunis hosen

Es handelt sich hier um eine sog. Skytala-Chiffre mit k=5
(Der Schlüssel k ist Anzahl der Zeilen)

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

- Zum Entschlüsseln: Anzahl Spalten = $\lceil$ Chiffretextlänge / k $\rceil$
- Anmerkung: Auch hier bleiben die Buchstabenhäufigkeiten erhalten

Buchstabenhäufigkeiten

Relative Häufigkeiten der Buchstaben in deutschen Texten

A	6,51%	N	9,78 %
B	1,89 %	O	2,51 %
C	3,06 %	P	0,79 %
D	5,08 %	Q	0,02 %
E	17,40 %	R	7,00 %
F	1,66 %	S	7,27 %
G	3,01 %	T	6,15 %
H	4,76 %	U	4,35 %
I	7,55 %	V	0,67 %
J	0,27 %	W	1,89 %
K	1,21 %	X	0,03 %
L	3,44 %	Y	0,04 %
M	2,53 %	Z	1,13 %

(Quelle: [Beut2015])

Buchstabenhäufigkeiten (Forts.)

Ein charakteristischer Wert: Koinzidenzindex I

$$I = \sum_{i=0}^{25} p_i^2$$

p_i ist relative Häufigkeit des i-ten Buchstabens

Koinzidenzindex im Deutschen

$$I = 0,0651^2 + 0,0189^2 + \ldots + 0,0113^2 = 0,0762$$

Ziel von Verschlüsselungsverfahren

... sollte es sein, dass der Koinzidenzindex des Chiffretextes minimal ist. Dann ist eine Frequenzanalyse unmöglich!

Welches ist der kleinstmögliche Wert für I?

Fundamentale Prinzpien von Verschlüsselungsverfahren

Konfusion

Verschleierung der Beziehung zwischen Schlüssel und Chiffretext, z. B. durch Substitutionsverfahren

Diffusion

Verbergen von statistischen Eigenschaften des Klartextes und damit Minimierung des Koinzidenzindexes, indem der Einfluss der Klartextzeichen auf möglichst viele Chiffretextzeichen gestreut wird

Modulare Arithmetik

Restklassenring $\mathbb{Z}_x$

- wir definieren $\mathbb{Z}_x = \{0, 1, \ldots, x-1\}$
- modulare Arithmetik ist das Rechnen mit einer begrenzten Menge an ganzen Zahlen
- Beispiel $\mathbb{Z}_{24}$ für die Uhrzeit:
  23 Uhr + 1 Stunde = 0 Uhr
  oder auch $23 + 1 \bmod 24 = 0$
  allerdings gilt z. B. auch $23 + 25 \bmod 24 = 0$
  sowie $23 - 47 \bmod 24 = 0$
- Berechnung des ganzzahligen Rests r:
  Seien $a, r, x \in \mathbb{Z}$ und $x > 0$
  Falls x ein Teiler von $a - r$ ist, schreibt man

$$a \equiv r \bmod x$$

„a und r sind kongruent bezgl. des Moduls x“

- für r gilt dann $a = q \cdot x + r$ wobei q eine ganze Zahl ist

Modulare Arithmetik (Forts.)

Beobachtung: Rest r ist nicht eindeutig

- für unterschiedliche Werte von q gibt es auch unterschiedliche Werte für r:
  $44 = 6 \cdot 7 + 2$
  $44 = 5 \cdot 7 + 9$
  $44 = 4 \cdot 7 + 16$
  $44 = 7 \cdot 7 - 5$ usw.
- die möglichen Reste bilden eine *Restklasse*:
  $$RK = \{ ..., -12, -5, 2, 9, 16, ... \}$$
- für alle $r \in RK$ gilt: $r = q \cdot 7 + 2$ mit $q \in \mathbb{Z}$
  (aber nur für $r = 2$ gilt $r \in \mathbb{Z}_7$)
- alle Elemente einer Restklasse verhalten sich äquivalent:
  Beispiel: Berechnung von $2^{10} \bmod 7 = 1024 \bmod 7$

$$\begin{aligned} 1024 &= 146 \cdot 7 + 2 \text{ (ist im Kopf nicht sooo einfach)} \\ \text{einfacher: } 1024 &= 100 \cdot 7 + 324 \\ 324 &= 50 \cdot 7 - 26 \\ -26 &= (-4) \cdot 7 + 2 \end{aligned}$$

$\Rightarrow$ wir können Rechnungen vereinfachen, ohne das Endergebnis zu ändern!

Modulare Arithmetik (Forts.)

Rechnen mit kleineren Zahlen

- es klappt auch mit der Multiplikation:
  $2^{10} = 2^2 \cdot 2^4 \cdot 2^4 = 4 \cdot 16 \cdot 16$
  $4 = 0 \cdot 7 + 4$
  $16 = 2 \cdot 7 + 2$
  also ist $2^{10} \bmod 7 = (4 \cdot 2 \cdot 2) \bmod 7 = 2$

Rechenoperationen im Restklassenring $\mathbb{Z}_x$

- Addition: $\boldsymbol{a} + \boldsymbol{b} \equiv \boldsymbol{c} \bmod \boldsymbol{x}$, wobei $c \in \mathbb{Z}_x$
- Multiplikation $\boldsymbol{a} \cdot \boldsymbol{b} \equiv \boldsymbol{d} \bmod \boldsymbol{x}$, wobei $d \in \mathbb{Z}_x$

Eigenschaften eines Rings

- abgeschlossen bzgl. Addition sowie Multiplikation
- Addition und Multiplikation sind assoziativ und kommutativ
- Existenz jeweils eines neutralen Elements bzgl. Addition sowie Multiplikation
- jedes Element hat eine additive Inverse

Modulare Arithmetik (Forts.)

Multiplikative Inverse im Restklassenring $\mathbb{Z}_x$

- $a \cdot a^{-1} \equiv 1 \bmod x \quad \Rightarrow a^{-1}$ heißt multiplikative Inverse zu a
- es existiert nicht zu jedem a eine multiplikative Inverse!
- a^{-1} die multiplikative Inverse zu a existiert nur dann, wenn $ggt(a, x) = 1$
- ihre Berechnung ist im Allgemeinen aufwändig

ggt – größter gemeinsamer Teiler (*engl.*: gcd)

- Berechnung mittels Primfaktorzerlegung; Beispiel:
  $540 = 2^2 \cdot 3^3 \cdot 5$
  $585 = 3^2 \cdot 5 \cdot 13$
  $\Rightarrow ggt(540, 585) = 3^2 \cdot 5 = 45$
- Berechnung auf diesem Weg meist aufwändig!
- $ggt(a, b) = 1$, falls a und b teilerfremd $\Rightarrow$ *relativ prim* oder *coprim*

Modulare Arithmetik (Forts.)

Rekursiver Algorithmus zur Berechnung des *ggt* (nach Euklid)

- für $a, b > 0$ gilt: $$ggt(a,b) = \begin{cases} b & \text{, falls } a = b \\ ggt(a-b,b) & \text{, falls } a > b \\ ggt(b-a,a) & \text{, falls } a < b \end{cases}$$
- Beispiel: $ggt(585,540) = ggt(45,540) = ggt(495,45) = ggt(450,45) = ggt(405,45) = \ldots = ggt(45,45) = 45$
- Schneller geht's meist mit:

  für $a, b > 0$ gilt: $$ggt(a,b) = \begin{cases} ggt(b,a) & \text{, falls } a < b \\ b & \text{, falls } a \bmod b = 0 \\ ggt(a \bmod b, b) & \text{, falls } a > b \end{cases}$$
- Beispiel: $ggt(585,540) = ggt(45,540) = ggt(540,45) = 45$

Aufgaben

(i) Berechnen Sie $136157 \cdot 10^{15} \bmod 11$

(ii) Berechnen Sie $ggt(237,27)$, $ggt(31,103)$ und $ggt(47,1)$

Affine Chiffren

Definition

- Wir rechnen im Zahlenring $\mathbb{Z}_x$
- Es sei $m, c, a, b \in \mathbb{Z}_x$:

  Verschlüsselung: $e_k(m) = c \equiv (a \cdot m + b) \bmod x$

  Entschlüsselung: $d_k(c) = m \equiv (a^{-1} \cdot (c - b)) \bmod x$

  mit dem Schlüssel $k = (a, b)$ mit $ggt(a, x) = 1$

  $\Rightarrow$ für a sind daher $\varphi(x)$ Werte möglich

  $\varphi(x)$ ist Euler-Funktion;
  ihr Wert ist die Anzahl der Zahlen in $\mathbb{Z}_x$, die teilerfremd zu x sind

Anmerkungen

- Welche Werte kommen in $\mathbb{Z}_{26}$ für a in Frage?
- Wie berechnet man zu a die multiplikative Inverse a^{-1}?

Polyalphabetische Chiffren

Vigenère-Chiffre (16. Jahrhundert)

- jedes Klartextzeichen kann auf jedes Chiffretextzeichen abgebildet werden
  $\Rightarrow$ Unterschied zu *monoalphabetischen Chiffren*
- Die Verschlüsselung des Klartextes $\boldsymbol{m} = (\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots \boldsymbol{m}_{\ell-1})$ der Länge ℓ wird mit Hilfe eines Schlüsselworts $\boldsymbol{k} = (\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_{n-1})$ der Länge $\boldsymbol{n}$ berechnet:

$$\boldsymbol{e_k}(\boldsymbol{m_i}) = (\boldsymbol{m_i} + \boldsymbol{k}_{\boldsymbol{i} \bmod \boldsymbol{n}}) \bmod 26 \quad \forall \boldsymbol{i} \in \{0, \ldots, \ell - 1\}$$

$$\boldsymbol{d_k}(\boldsymbol{c_i}) = (\boldsymbol{c_i} - \boldsymbol{k}_{\boldsymbol{i} \bmod \boldsymbol{n}}) \bmod 26 \quad \forall \boldsymbol{i} \in \{0, \ldots, \ell - 1\}$$

(für das Alphabet mit 26 Buchstaben)

Beispiel

- $\boldsymbol{m} = \boldsymbol{ANGRIFFUMSECHS}$, $\boldsymbol{k} = \boldsymbol{IMHO}$

m:	(0	13	6	17	8	5	5	20	12	18	4	2	7	18)
k:	(8	12	7	14	8	12	7	14	8	12	7	14	8	12)
c:	(8	25	13	5	16	17	12	8	20	4	11	16	15	4)

$\Rightarrow \boldsymbol{c} = \boldsymbol{iznfqrmiuelqpe}$

Vigenère-Chiffre (Forts.)

Kryptoanalyse

(1) Bestimmen der Schlüsselwortlänge n für einen Chiffretext der Länge ℓ

- *Kasiski-Test*:
  finde mehrere gleiche Folgen im Chiffretext und bestimme deren Abstand x_i
  $\Rightarrow$ Vermutung: $x_i = a_i \cdot n$ mit $a_i \in \mathbb{Z}$ ($\Rightarrow$ ggt berechnen)
- *Friedman-Test*:
  berechne den Koinzidenzindex I des Chiffretextes und damit

$$h = \frac{0,0377\ell}{I \cdot (\ell - 1) - 0,0385\ell + 0,0762}$$

$\Rightarrow$ h liegt in der Größenordnung von n (Herleitung vgl. [Beut2015])

(2) spaltenweise Dechiffrierung mittels Frequenzanalyse

$\Rightarrow$ Dechiffrierung ist deutlich einfacher, wenn Teile des Klartextes bekannt sind oder erraten werden können

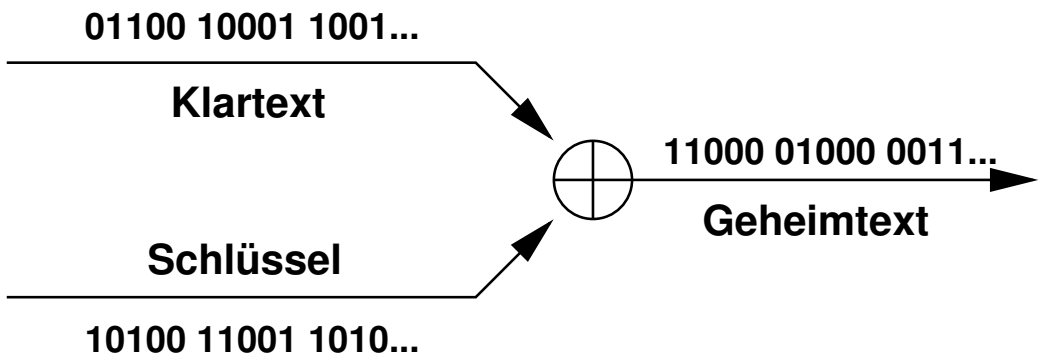
Sicherheit

Die Sicherheit ist umso größer, je kürzer der Klartext und je länger der Schlüssel!

One-Time Pad (OTP)

Prinzip einer perfekt sicheren Verschlüsselung

- Wähle einen Schlüssel ebenso lang wie der Text.
- Wähle jedes einzelne Schlüsselbit echt zufällig.
- Verwende jeden Schlüssel immer nur einmal.



Frage: *Wieso ist das perfekt sicher?*

One-Time-Pad (Forts.)

Anforderung an ein perfekt sicheres Kryptosystem

- Durch Untersuchen des Chiffretextes lassen sich keine Informationen über den Klartext ableiten
- Die Wahrscheinlichkeit, dass der Klartext $\boldsymbol{m}$ gesendet wurde, wenn der Chiffretext $\boldsymbol{c}$ empfangen wurde, ist gleich der Wahrscheinlichkeit, dass $\boldsymbol{m}$ gesendet wurde

$$\Rightarrow \boldsymbol{Prob}(\boldsymbol{m}|\boldsymbol{c}) = \boldsymbol{Prob}(\boldsymbol{m}) \quad \forall \boldsymbol{m} \in \boldsymbol{P}, \boldsymbol{c} \in \boldsymbol{C}$$

- Das OTP ist ein perfekt sicheres Kryptosystem
  Beweis: Shannon 1945 (Argumentation über Wahrscheinlichkeitsverteilung)
- Durch die bitweise XOR-Verknüpfung ist der Chiffretext genauso zufällig wie der Schlüssel
- Das OTP kann als Sonderfall der Vigenère-Chiffre angesehen werden

One-Time-Pad (Forts.)

Probleme in der Praxis

- Wie werden die Schlüssel sicher ausgetauscht?
  (Schlüssel sind i. Allg. sehr lang!)

  $\Rightarrow$ Keine spontane Kommunikation möglich!

- Der Schlüssel muss echt zufällig sein!
  $\Rightarrow$ Wie generiert man kryptografisch sichere (echte) Zufallszahlen?
  (nicht trivial, aber möglich)

- Schlüssel darf sich nicht wiederholen:

  J. Mason et. al., 2006:

  A Natural Language Approach to Automated Cryptanalysis of Two-time Pads (published at CCS'06)

Krypto-Kontroverse

Kryptografie für alle?

- Verwendung kryptografischer Verfahren für Privatpersonen war in Frankreich bis 1998 verboten
- Bis 2000 war die Ausfuhr **starker** krypt. Verfahren aus den USA strafbar
  $\Rightarrow$ Crypto Wars
  (vgl. `des`-Befehl bei Unix, Zimmermann-Prozess, Clipper-Chip, ...)
- 19. Oktober 2000:
  Lockerung der US-Exportbeschränkungen,
  insbesondere in die EU

Frage: *Wie ist es heute – was spricht für, was gegen ein Kryptoverbot?*

Zufallszahlen

Arten von Zufallszahlen

- Echte Zufallszahlen
  - Generierung mittels TRNG (*True Random Number Generator*)
  - basieren i. Allg. auf physikalischen Zufallsprozessen
    (z. B. radioaktiver Zerfall oder Spannungsabfall an einer Diode)

- Pseudozufallszahlen
  - Generierung mittels PRNG (*Pseudo Random Number Generator*)
  - besitzen statistische Eigenschaften echter Zufallszahlen
  - folgen einer deterministischen Funktion
    $\Rightarrow$ sind reproduzierbar
  - Einsatz bei Tests, Simulationen, Spielen
  - i. Allg. unbrauchbar für kryptografische Anwendungen

Pseudozufallszahlen

PRNG:Lineare Kongruenzmethode (Lehmer, 1949)

Berechnung der Pseudozufallszahlenfolge $\boldsymbol{X}$ für einen Modulus $\boldsymbol{m}$ mittels

$$\boldsymbol{X_{n+1}} = (\boldsymbol{a} \cdot \boldsymbol{X_n} + \boldsymbol{c}) \bmod \boldsymbol{m}$$

mit $\boldsymbol{X_0} : 0 \leq \boldsymbol{X_0} < \boldsymbol{m}$ als *Seed* (Startwert)
und $\boldsymbol{a}, \boldsymbol{c} : 0 \leq \boldsymbol{a}, \boldsymbol{c} < \boldsymbol{m}$

Beispiele

- $\boldsymbol{m} = 10, \boldsymbol{X_0} = \boldsymbol{a} = \boldsymbol{c} = 7$
  $\Rightarrow \boldsymbol{X}$: 7, 6, 9, 0, 7, 6, 9, 0 , ...
  (Periodenlänge 4)

- $\boldsymbol{m} = 17, \boldsymbol{X_0} = 1, \boldsymbol{a} = 7, \boldsymbol{c} = 0$
  $\Rightarrow \boldsymbol{X}$: 1, 7, 15, 3, 4, 11, 9, 12, 16, 10, 2, 14, 13, 6, 8, 5, 1, 7, ...
  (Periodenlänge 16)

Pseudozufallszahlen (Forts.)

Anforderungen an einen PRNG

- maximale Periodenlänge $m - 1$
  $\Rightarrow$ m muss eine Primzahl sein sowie passende Wahl von a und c
- möglichst großer Modulus m
  $\Rightarrow$ bei 32-Bit Systemen z. B. $m = 2^{31} - 1$, $a = 16\,807, c = 0$
- die erzeugte Folge sollte zufällig aussehen
  $\Rightarrow$ statistische Tests

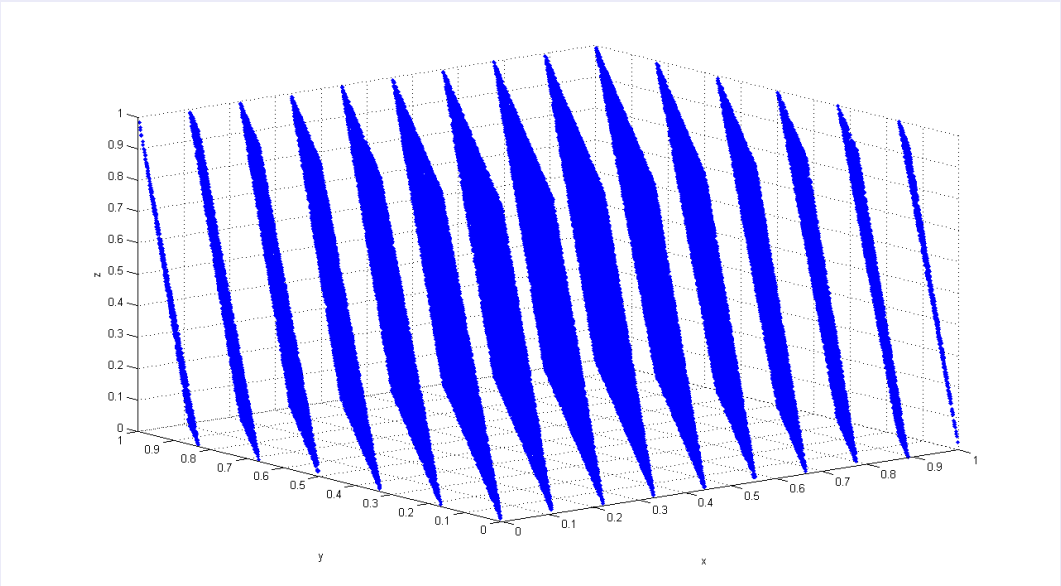
Qualität der Zufallszahlen

- PRNG sollte gleichverteilte (rechteckverteilte) Pseudozufallszahlen generieren
- mittels geeigneter Transformationen lassen sich Pseudozufallszahlen einer beliebigen statistischen Verteilung erzeugen
- Zahlenfolgen lassen sich reproduzieren, wenn derselbe Seed verwendet wird (wichtig für Simulationen)
- nachfolgende Zahlen lassen sich vorhersagen, wenn die Parameter m, a, c bekannt sind und ein X_n ($\Rightarrow$ unbrauchbar für die Kryptografie!)

Pseudozufallszahlen (Forts.)

Spektraltest

- Test, ob ein RNG gleichmäßig verteilte Zufallszahlen erzeugt
- jeweils $\boldsymbol{n}$ aufeinanderfolgende Zufallszahlen werden als ein $\boldsymbol{n}$-Tupel betrachtet
- liegen diese auf einer begrenzten Anzahl von Hyper-Ebenen, so ist deren Verteilung nicht gleichmäßig und der RNG von minderwertiger Qualität
- Beispiel: RANDU ($\boldsymbol{m} = 2^{31} - 1; \boldsymbol{a} = 65539 = 2^{16} + 3; \boldsymbol{c} = 0; \boldsymbol{X}_0 = 1$), $\boldsymbol{n} = 3$



Quelle: Der ursprünglich hochladende Benutzer war Luis Sanchez in der Wikipedia auf Englisch – Übertragen aus en.wikipedia nach Commons durch sevela.p. Dieses Diagramm wurde mit MATLAB erstellt, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3832343

Pseudozufallszahlen (Forts.)

Run-Test *

- In einem (pseudo-)zufällig erzeugten Bitstrom der Länge n werden Folgen von gleichen Bits identifiziert (sog. Runs) und deren Anzahl bestimmt.
- Mit Hilfe der Statistik lässt sich zeigen, dass der Erwartungswert bei $E(R) = \frac{n+1}{2}$ liegt, wobei die Zufallsvariable R die Anzahl der Runs in einem echt zufälligen Bitstrom beschreibt.
- Weicht die Anzahl der Runs des untersuchten Bitstroms (hier: a) signifikant hiervon ab, wurde dieser vermutlich mittels eines minderwertigen PRNG erzeugt.

$$Prob(R - 1 = a) = \binom{n-1}{a-1} \cdot 0,5^{a-1} \cdot 0,5^{n-a}$$

Weitere Testmethoden

- Poker-Test
- Chi-Quadrat-Test
- ...

* Knuth, D.E. (1981): The Art of Computer Programming, Vol. 2 Seminumerical Algorithms, 2. ed. Addison-Wesley, Reading, Mass.

Pseudozufallszahlen (Forts.)

Kryptografisch sichere PRNG's

- es wird ein Strom zufälliger Bits erzeugt
- die erzeugte Folge muss alle bekannten statistischen Tests bestehen
- es ist rechnerisch unmöglich, das nächste Bit vorherzusagen – auch bei Kenntnis des Algorithmus und aller zuvor erzeugten Bits
- die zufällige Bitfolge ist nicht reproduzierbar

$\Rightarrow$ hierzu werden häufig kryptografisch starke Verschlüsselungsalgorithmen eingesetzt (wegen Diffusion und Konfusion)

Beispiel:

$$X_n = e_{MK}(n) \quad \forall n \in \mathbb{Z}_x$$

wobei MK ein geheimer Master-Key (z. B. für AES) und $log_2(x)$ die Blocklänge des Verschlüsselungsalgorithmus ist

Kryptografisch sichere Pseudozufallszahlen (Forts.)

Beispiel: Blum-Blum-Shub (BBS) CSRNG

- kryptografisch sicher unter der Annahme, dass die Faktorisierung ein hartes Problem ist
- Algorithmus für eine pseudozufällige Bitfolge z der Länge ℓ:
  (1) Wähle zufällig $p \neq q$ als Primzahlen mit $p \equiv 3 \bmod 4$ und $q \equiv 3 \bmod 4$
  $\Rightarrow n = p \cdot q$
  (2) Wähle zufällig $s \in \{2, 3, \ldots, n-1\}$, so dass $ggt(s, n) = 1$
  (3) $x_0 = s^2 \bmod n$
  (4) $for(i = 1; i \leq \ell; i++)$ {
  $x_i = x_{i-1}^2 \bmod n$
  $z_i = x_i \bmod 2$
  }

Echte Zufallszahlen

TRNG's

- Echter Zufall entsteht in physikalischen Prozessen
  $\Rightarrow$ spezielle Hardware erforderlich
  - Münzwurf
  - Ziehung der Lottozahlen
  - Anzahl der zerfallenen Atome beim radioaktiven Zerfall
  - Rauschen in einer Fotodiode (allg.: quantenmechanische Prozesse)
  - Uhrendrift (Abweichung von Uhren von der Realzeit)

- oder kann durch Beobachtung zufälliger interner Abläufe in einem Computer und sog. Randomness Extraction abgeleitet werden
  ($\Rightarrow$ Vorsicht, das könnte ein Angreifer evtl. beeinflussen!)
  - CPU-Burst (Zeit bis zum Prozesswechsel)
  - Zeit zwischen Tastaturanschlägen
  - Ausgabewerte einer Computermaus
  - ...

  $\Rightarrow$ sammeln durch das Betriebssystem in einem Entropie-Pool,
  z. B. `/dev/random` bei Linux

DES (Data Encryption Standard)

Entwicklung

- Ursprünglich von IBM: Lucifer-Chiffre, Anpassungen durch NSA
- (US-)Standard für Finanzen und Kommunikation (1977 – 2002)

Vorzüge

- einfache Basisoperationen (Permutation, Substitution, Shift, XOR)
- leicht implementierbar in Hardware
- schnell (bis zu 1 GB/sec)

Sicherheit

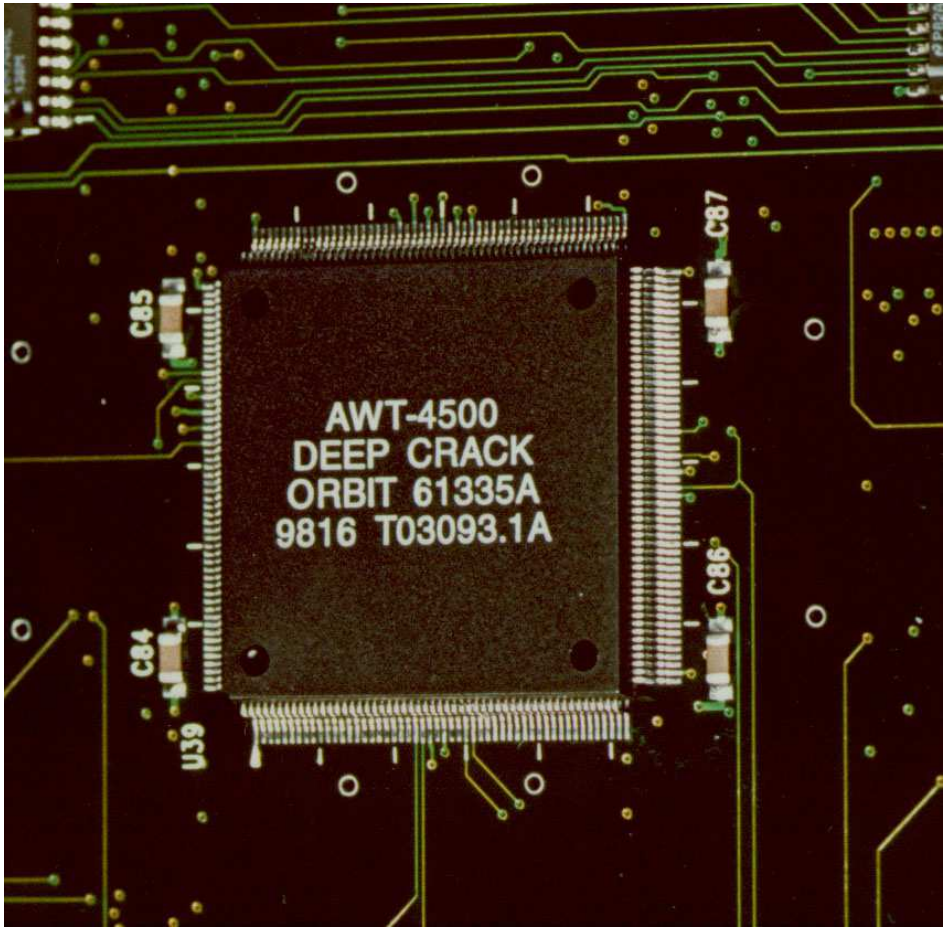
- Blockchiffre mit Blocklänge: 64 Bit (heute zu kurz)
- Schlüssellänge: 64 Bit, effektiv 56 Bit (heute zu kurz)
- Brute-Force Angriff im Jahr 1999: 22 h 15 min (*Deep Crack*, EFF)
- Seit 2004 wird Einsatz vom NIST offiziell nicht mehr empfohlen.

DES – Deep Crack

- Entwickelt und gebaut von der EFF (DES-Challenge III, 1999)
- Kosten damals ca. 250 000 USD
- Leistungsfähigkeit:
  ca. $90 \cdot 10^9$ Schlüssel / Sekunde

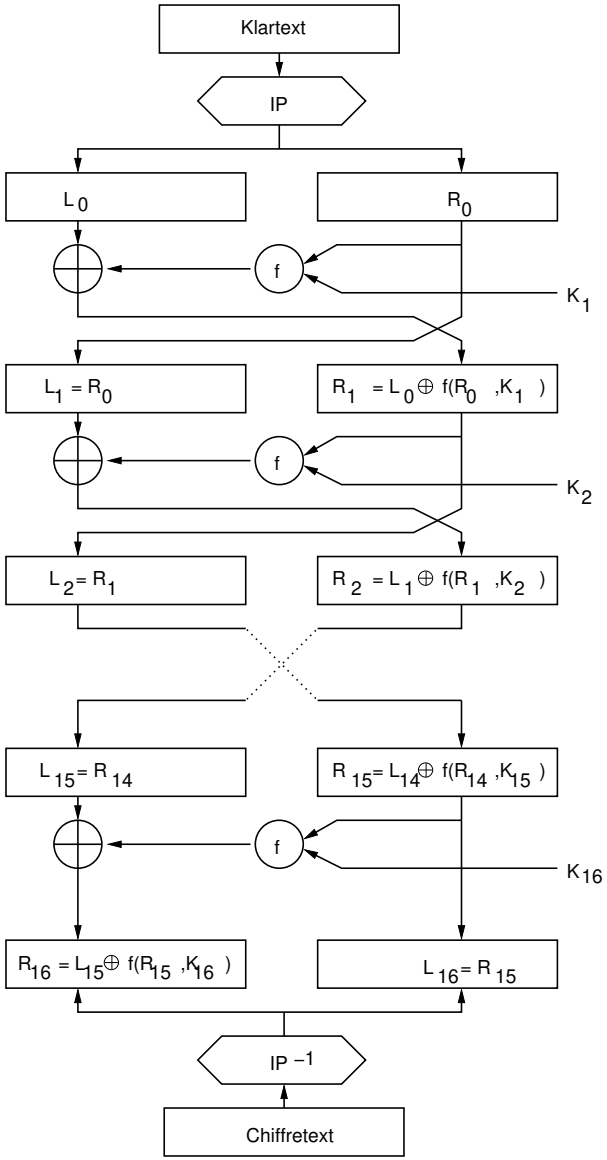
Durchschnittliche Zeit zum Knacken einer DES-Verschlüsselung: ca. 5 Tage

DES – Deep Crack (Forts.)



- Speziell entworfener DES-Chip
- 29 Boards insgesamt im Deep Crack
- 64 Chips pro Board

DES (Forts.)


Klartext
IP
L 0
R0
f
K1
L1 = R0
R1 = L0 ⊕ f(R0 ,K1)
f
K2
L2= R1
R 2 = L 1 ⊕ f(R 1 ,K2)
L 15= R 14
R 15= L14⊕ f(R14 ,K15)
f
K16
R16 = L15⊕ f(R15 ,K16)
L 16= R 15
IP −1
Chiffretext

DES (Forts.)

Interner Aufbau

- 16 Runden mit identischem Aufbau aber unterschiedlichen Rundenschlüsseln
- Rundenschlüssel werden in jeder Runde neu vom Hauptschlüssel abgeleitet und dabei auf 48 Bit verkürzt
- Eingangs- und Abschlusspermutation
- pro Runde wird jeweils nur die linke Hälfte des Blocks mit Funktion f verschlüsselt
- Idee: aus der linken Hälfte eines Blocks wird mittels f eine Pseudozufallszahl errechnet, die als Schlüssel verwendet wird, um mittels der XOR-Funktion die rechte Hälfte zu verschlüsseln
- nur Funktion f mit kryptografischer Relevanz
- Ver- und Entschlüsselung sind symmetrisch zueinander

DES (Forts.)

Entschlüsselung

Wieso sind Verschlüsselungs- und Entschlüsselungoperation pro Runde identisch?

- Wir betrachten die Verschlüsselung der letzten Runde mit k_{16}:

  m=$\boxed{L_{15}|R_{15}}$ $\Rightarrow$ c=$\boxed{L_{16}|R_{16}}$

  genauer: $L_{16} = R_{15}$ und $R_{16} = L_{15} \oplus f(R_{15}, k_{16})$

- Entschlüsseln durch erneutes Verschlüsseln mit k_{16}:

  c=$\boxed{L_{16}|R_{16}}$ $\Rightarrow$ m'=$\boxed{L'_{15}|R'_{15}}$

  (1) Wir setzen $a = L_{16}, b = R_{16}$

  (2) $R'_{15} = a = L_{16} = R_{15}$

  (3) $L'_{15} = b \oplus f(a, k_{16}) = R_{16} \oplus f(R_{15}, k_{16})$
  $= [L_{15} \oplus f(R_{15}, k_{16})] \oplus f(R_{15}, k_{16}) = L_{15}$

DES (Forts.)

Funktion f

- f realisiert Konfusion (mittels Substitution) und Diffusion (mittels Permutation):
  - ändert sich ein Eingangsbit, dann ändern sich ca. die Hälfte der Ausgangsbits
  - ändert sich ein Schlüsselbit, dann ändern sich viele Ausgangsbits
- Eingabe: 32 Bit Daten (m_{32}) sowie 48 Bit Rundenschlüssel (k)
- Schritte:
  (1) m_{32} wird mittels Expansion auf 48 Bit erweitert ($\Rightarrow m_{48}$)
  (2) $m' = m_{48} \oplus k$
  (3) m' wird in acht Blöcke zu je 6 Bit unterteilt
  (4) jeder 6-Bit-Block wird mittels einer individuellen S-Box ($S_1, \ldots, S_8$) zu einem 4-Bit-Block substitutiert
  (5) die resultierenden acht Blöcke werden zu einem 32-Bit-Block zusammengefasst und die Bits abschließend noch einmal permutiert

DES (Forts.)

Expansion

Eingabe: $\boldsymbol{m}_{32}$
Ausgabe: $\boldsymbol{m}_{48}$

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES (Forts.)

S-Boxen

Beispiel: S-Box $\boldsymbol{S}_1$

$\boldsymbol{S}_1$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	**09**	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

(Die weiteren S-Boxen finden Sie z. B. in [PP2016])

Substitution

- Beispiel: Eingabe 52 $\Rightarrow$ | **1** | 1 | 0 | 1 | 0 | **0** |
- das erste und das letzte Bit (1; 0) bestimmen die Zeile $\Rightarrow$ Zeile 2
- die Bits dazwischen (1; 0; 1; 0) die Spalte $\Rightarrow$ Spalte 10
- nach S-Box $\boldsymbol{S}_1$ ist die Ausgabe: $09 = 1001_2$

DES (Forts.)

S-Boxen (Forts.)

- die S-Boxen sind sehr sorgfältig festgelegt und garantieren die Sicherheit des DES
- sie implementieren eine nicht-lineare Funktion, d. h. es gilt:

$$\boldsymbol{S}(\boldsymbol{a}) \oplus \boldsymbol{S}(\boldsymbol{b}) \neq \boldsymbol{S}(\boldsymbol{a} \oplus \boldsymbol{b})$$

  $\Rightarrow$ ein Zusammenhang zwischen Eingang und Ausgang lässt sich **nicht** mittels eines linearen Gleichungssystems beschreiben!
- darüberhinaus bieten sie Schutz gegen die sog. *differenzielle Kryptoanalyse**

* *chosen plaintext attack* auf rundenbasierte Blockchiffren (Biham, Shamir 1991): Wie wirken sich Differenzen im Klartext auf den Chiffretext aus?

DES (Forts.)

Besondere Eigenschaften

- Wenn wir von $\boldsymbol{K}$ und $\boldsymbol{m}$ das bitweise Komplement bilden, ergibt sich auch das Komplement des Chiffretextes:

$$\overline{\boldsymbol{DES}_{\boldsymbol{K}}(\boldsymbol{m})} = \boldsymbol{DES}_{\overline{\boldsymbol{K}}}(\overline{\boldsymbol{m}})$$

- Es existieren vier schwache Schlüssel, die man nicht verwenden sollte:
  - $\boldsymbol{K}_1 = (0\ldots0\ 0\ldots0)$
  - $\boldsymbol{K}_2 = (0\ldots0\ 1\ldots1)$
  - $\boldsymbol{K}_3 = (1\ldots1\ 0\ldots0)$
  - $\boldsymbol{K}_4 = (1\ldots1\ 1\ldots1)$
  - für diese vier Schlüssel gilt: $\boldsymbol{DES}_{\boldsymbol{K}}(\boldsymbol{m}) = \boldsymbol{DES}_{\boldsymbol{K}}^{-1}(\boldsymbol{m}) \quad \forall \boldsymbol{m}$

Mehrfachverschlüsselungen

Sicherheit

- keine Verbesserung bei vielen Verschlüsselungsalgorithmen
  $\Rightarrow$ Beispiel: Verschiebe-Chiffre

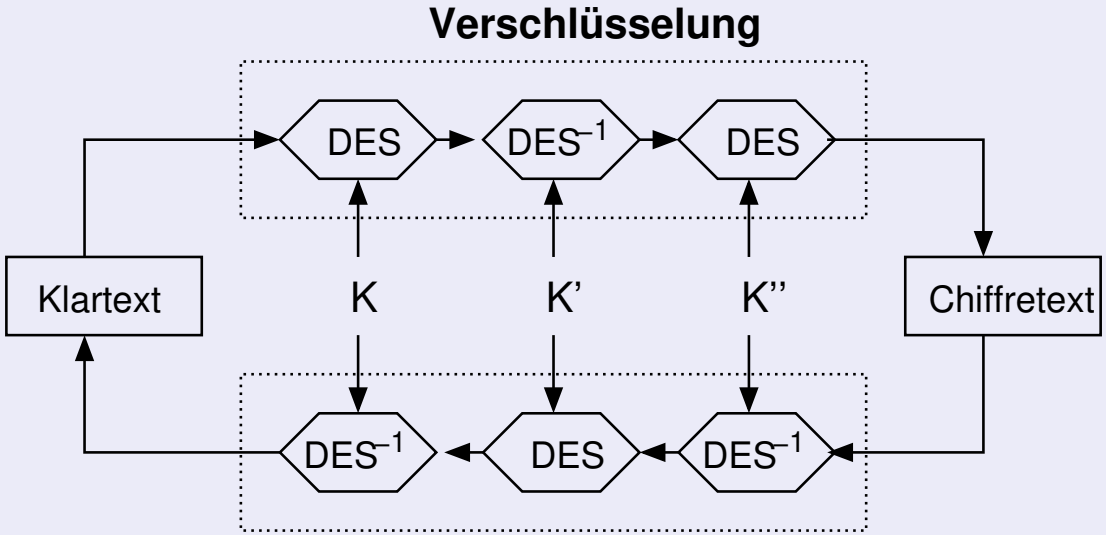
Mehrfachverschlüsselung beim DES

- Zweifachverschlüsselung mit unterschiedlichen Schlüsseln
  - Angreifer muss zwei DES-Schlüssel $\boldsymbol{K}_1$ und $\boldsymbol{K}_2$ erraten
    $\Rightarrow$ effektive Schlüssellänge $= 2 \cdot 56 = 112$ Bit
  - Meet-in-the-middle Angriff vereinfacht den Angriff, wenn ein Klartext $\boldsymbol{m}$ und der Chiffretext $\boldsymbol{c}$ bekannt sind
    - Angreifer verschlüsselt $\boldsymbol{m}$ mit allen 2^{56} möglichen Schlüsseln $\boldsymbol{K}_1$ $\Rightarrow$ Menge $\boldsymbol{X}$
    - Angreifer entschlüsselt $\boldsymbol{c}$ mit allen 2^{56} möglichen Schlüsseln $\boldsymbol{K}_2$ $\Rightarrow$ Menge $\boldsymbol{Y}$
    - Angreifer sucht Übereinstimmung in $\boldsymbol{X}$ und $\boldsymbol{Y}$
      $\Rightarrow$ Gesamtaufwand $= 2^{56} + 2^{56}$

    $\Rightarrow$ keine signifikante Erhöhung des Sicherheitsniveaus!

DES-Variante

Tripple DES (3DES, TDES)



- Vorteile:
  - Schlüssellänge 168 Bit, effektiv 112 Bit (wegen möglichem Meet-in-the-middle Angriff)
  - einfache Realisierung, bewährter Algorithmus
  - kompatibel zu DES, falls $K = K' = K''$
- Nachteile:
  - dreifache Verschlüsselungszeit
  - nur 64 Bit Blocklänge $\Rightarrow$ gilt seit 2016 als unsicher[1]

[1] K. Bhargavan, G. Leurent: On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. ACM CCS 2016 - 23rd, pp.456-467

AES (Advanced Encryption Standard)

Wissenswertes

- nach Ausschreibung des NIST seit dem Jahr 2002 (US-)Standard
- in den USA zugelassen für Regierungskommunikation bis hin zu vertraulichen Dokumenten der Sicherheitsstufe TOP SECRET
- 128, 192, 256 Bit Schlüssellängen aber nur 128 Bit Blocklänge
- identisch zum Rijndael-Algorithmus (Joan Daemen und Vincent Rijmen 1998)
  - effizient in Hardware und Software implementierbar
  - 128, 192, 256 Bit Schlüssel- und Blocklängen
  - potienziell erweiterbar auf $32 \cdot x$ Bits

Weitere finale Kandidaten damals

- MARS (IBM)
- RC6 (RSA)
- Serpent (Anderson et. al.)
- Twofish (Schneier et. al.)

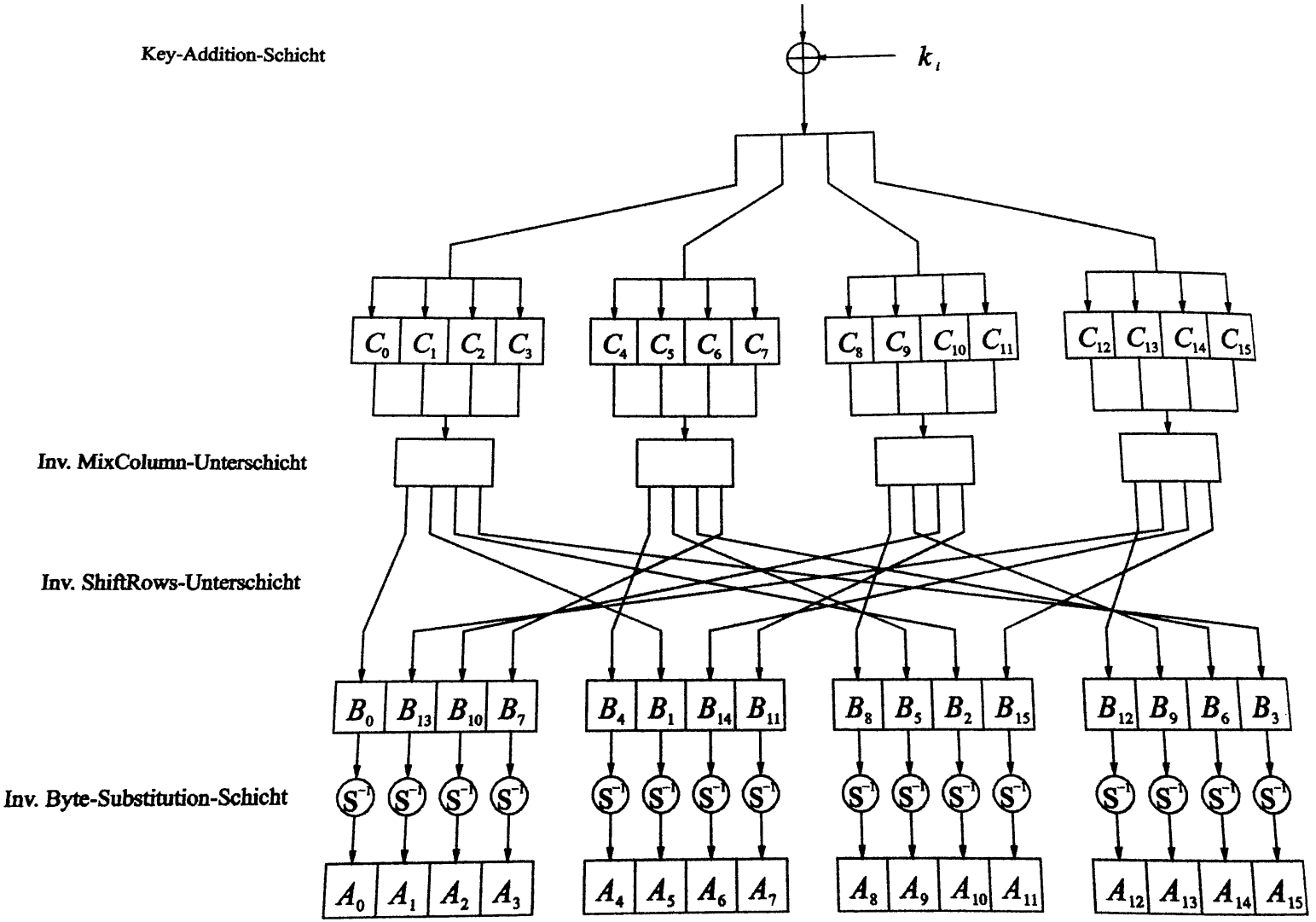
AES (Forts.)

Interne Struktur

- je nach Schlüssellänge 10, 12 bzw. 14 interne Runden
- Rundenschlüssel werden mittels Transformation aus Hauptschlüssel abgeleitet
- jede Runde setzt sich aus mehreren Schichten (Layers) zusammen
  - **Key Addition Layer:**
    bitweise XOR Verknüpfung des 128 Bit Rundenschlüssels mit den 128 Bit Rundendaten
  - **Byte Substitution Layer:**
    nichtlineare Transformation der Rundendaten mittels S-Boxen
  - **Diffusion Layer**:
    byteweise Permutation der Daten (*ShiftRows*) und Matrixmultiplikation (*MixColumn*)
- Details finden sich z. B. in [PP2016]

AES (Forts.)

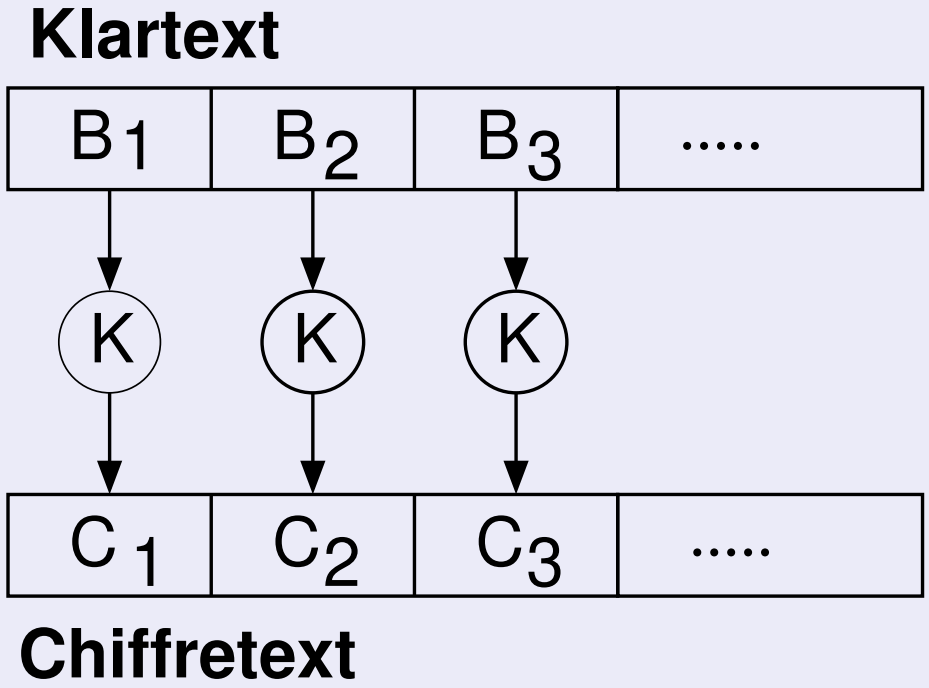
Schematischer Aufbau einer AES-Runde



(Quelle: [PP2016])

Verschlüsselungsbetriebsarten für Blockchiffren

ECB: Electronic Code Book

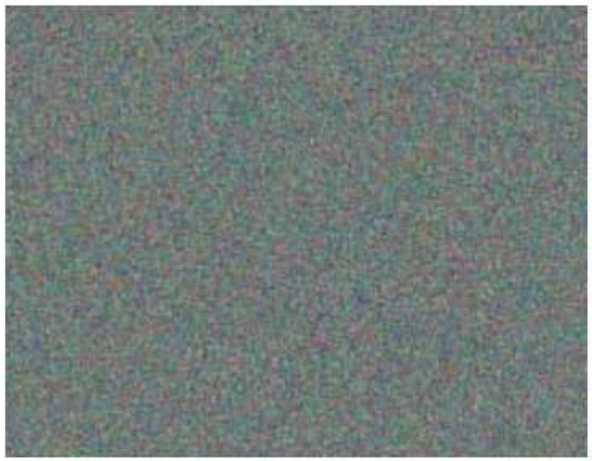


- Vorteile:
  Fehlertoleranz bei verrauschtem Kanal, mögliche Parallelität
- Nachteile:
  viele Angriffsmöglichkeiten, insb. kein Schutz der Integrität, Strukturen im Klartext bleiben erhalten (insb. bei Bildern)

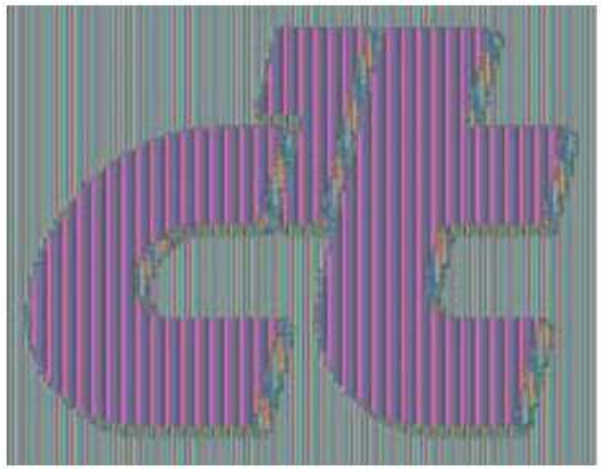
Verschlüsselungsbetriebsarten (Forts.)

Probleme mit ECB:

(a) Original



(b) CBC-AES-Verschlüsselung

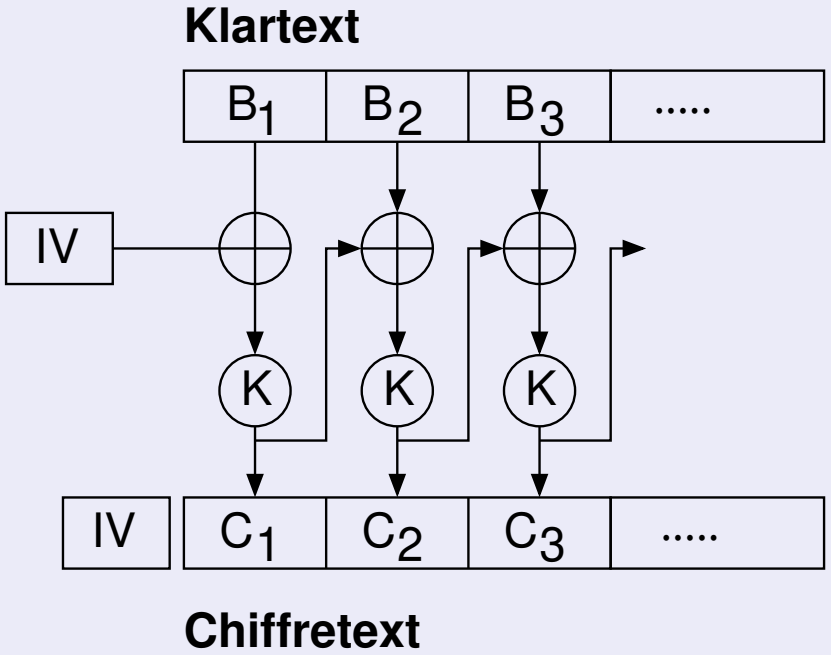


(c) ECB-AES-Verschlüsselung

(Quelle: http://heise.de/-3221002)

Verschlüsselungsbetriebsarten (Forts.)

CBC: Cipher Block Chaining



- Vorteil:
  bessere Sicherheit als ECB
- Nachteile:
  schlechte Fehlertoleranz, keine parallele Verschlüsselung,
  Denial-of-Service- und Padding-Attacken
- Alternativen: Cipher Feed Back (CFB), Output Feed Back (OFB)

Verschlüsselungsbetriebsarten (Forts.)

CTR: Counter Mode

Der Chiffretextblock c_i berechnet sich aus dem Klartextblock m_i

- mit Hilfe des zufällig gewählten Initialisierungsvektors IV ($\Rightarrow$ *nonce*),
- eines wechselnden Counter-Werts (z. B. inkrementelle Integer-Zahlen) und
- der Verschlüsselungsfunktion e (z. B. AES) mit Schlüssel k

zu:

$$c_i = m_i \oplus e_k(IV \circ ctr_i) \qquad (\circ \text{ ist die Konkatenation})$$

- Beispiel AES-128: IV der Länge 96 Bit, ctr der Länge 32 Bit
- Vorteile:
  hohe Sicherheit (bei guter Chiffre), parallele Ver- / Entschlüsselung
  $\Rightarrow$ schnell, wahlfreier Blockzugriff, Fehlertoleranz bei Bitfehlern
- Nachteile:
  kein Schutz der Integrität, einige Counter-Funktionen weniger sicher,
  XOR-Wiederholungsproblematik, IV darf nicht wiederverwendet werden

Verschlüsselungsbetriebsarten (Forts.)

GCM: Galois Counter Mode

- authentisierte Verschlüsselung
  $\Rightarrow$ berechnet c_i im CTR-Modus und zusätzlich MAC
- Verschlüsselung:
  $$c_i = e_k(ctr_i) \oplus m_i \quad \forall i \geq 1$$
  ctr_0 wird aus IV und einer Seriennummer abgeleitet; $ctr_i = ctr_{i-1} + 1$
- MAC (Message Authentication Code): kryptografische Prüfsumme
  - unter Verwendung von AAD (Additional Authenticated Data)
  - Algorithmus: (alle Multiplikationen finden mod $P(x)$ im endlichen Körper $GF(2^{128})$ mit $P(x) = x^{128} + x^7 + x^2 + x + 1$ statt)

  (1) $H = e_k(0)$

  (2) $g_0 = AAD \cdot H$

  (3) $g_i = (g_{i-1} \oplus c_i) \cdot H \quad \forall i \in \{1, \ldots, n\}$

  (4) $MAC = (g_n \cdot H) \oplus e_k(ctr_0)$

Frage: *Wie wird die Nachricht beim Empfänger geprüft?*

Rechnen im endlichen Erweiterungskörper $GF(2^m)$

Irreduzibles Polynom

- Ein Polynom $P(x)$ heißt **irreduzibel** (oder auch prim) , falls es sich **nicht** als Produkt (in $GF(2)$) zweier anderer Polynome darstellen lässt

  $\Rightarrow$ vgl. Generatorpolynom beim CRC
- Beispiel: $x^4 + x^3 + x + 1 = (x^2 + x + 1)(x^2 + 1)$ ist nicht irreduzibel

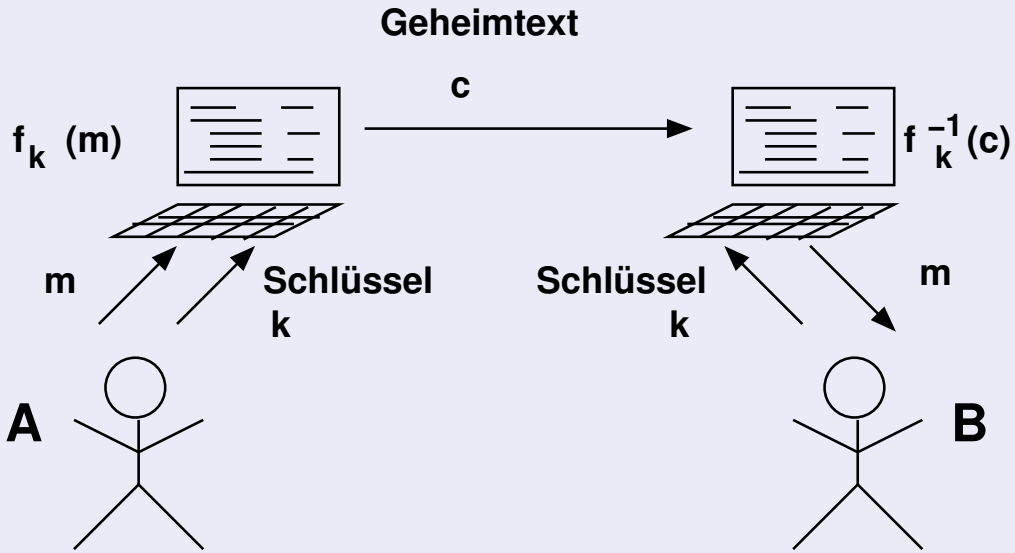
Multiplikation in $GF(2^m)$

- Sei $P(x)$ ein irreduzibles Polynom
- $a(x) \cdot b(x) \equiv r(x) \bmod P(x)$

  $\Rightarrow$ $r(x)$ ist der Rest, der entsteht, wenn wir das Produkt $a(x)b(x)$ durch $P(x)$ teilen (in $GF(2)$)
- Beispiel: (mit Bit-Strings)
  $a(x) = x^3 + x^2 + 1 \mathrel{\widehat{=}} 1101 = a$, $b(x) = x^2 + x \mathrel{\widehat{=}} 110 = b$,
  $P(x) = x^4 + x + 1 \mathrel{\widehat{=}} 10011 = P$
  $\Rightarrow ab = 101110$; $ab \bmod P \equiv 1000 = r \Rightarrow a(x) \cdot b(x) \equiv x^3$

Das Schlüsseltausch-Problem

Problem bei symmetrischer Verschlüsselung



Frage: *Wie erfährt Bob den Schlüssel **K** zum Dechiffrieren?*

Das Schlüsseltausch-Problem (Forts.)

Idee (?)

- Übertragung über anderen, „sicheren“ Kanal
- Welcher Kanal ist sicher? Was bedeutet hier “sicher”?

$\Rightarrow$ I.Allg. umständlich, nicht praktikabel!

Eleganter

- Wir nutzen den (unsicheren) Nachrichtenkanal.
- **Problem:**
  Der Schlüssel darf nicht im Klartext übertragen werden!

Frage: *Kann man den Schlüssel nicht verschlüsseln?*

Das Schlüsseltausch-Problem (Forts.)

Idee: Schlüssel wird gemeinsam konstruiert

- A und B überlegen sich eine geheime Zahl.
- Aus dieser Zahl berechnete Werte werden dem anderen mitgeteilt.
- Aus den geheimen und bekanntgegebenen Werten wird der Schlüssel berechnet.

Protokollentwurf

- A und B kennen beide eine Zahl $\boldsymbol{s}$ und eine Funktion $\boldsymbol{F}()$.
- A wählt $\boldsymbol{a}$, berechnet $\boldsymbol{F}(\boldsymbol{a}, \boldsymbol{s})$, sendet $\boldsymbol{F}(\boldsymbol{a}, \boldsymbol{s})$ an B.
  B wählt $\boldsymbol{b}$, berechnet $\boldsymbol{F}(\boldsymbol{b}, \boldsymbol{s})$, sendet $\boldsymbol{F}(\boldsymbol{b}, \boldsymbol{s})$ an A.
- A berechnet $\boldsymbol{K} = \boldsymbol{F}(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{b}))$, B berechnet $\boldsymbol{K} = \boldsymbol{F}(\boldsymbol{b}, \boldsymbol{F}(\boldsymbol{a}))$.

Frage: *Welche Eigenschaften muss $\boldsymbol{F}()$ haben?*

Das Schlüsseltausch-Problem (Forts.)

Zur Funktion $F()$

- Angriff:
  - Angreifer M belauscht $F(a, s)$ und $F(b, s)$.
  - M kennt $F()$ und s (wurden öffentlich bekanntgemacht).
  - M berechnet $K = F(F(a, s), F^{-1}(F(b, s), s))$.
- Abwehr des Angriffs:
  - Wähle eine Funktion $F()$, für die gilt:
    F() ist rechentechnisch einfach zu berechnen, jedoch $F^{-1}()$ nicht.
    $\Rightarrow$ $F()$ heißt *Einwegfunktion*.
  - Problem: Finde eine Einwegfunktion!

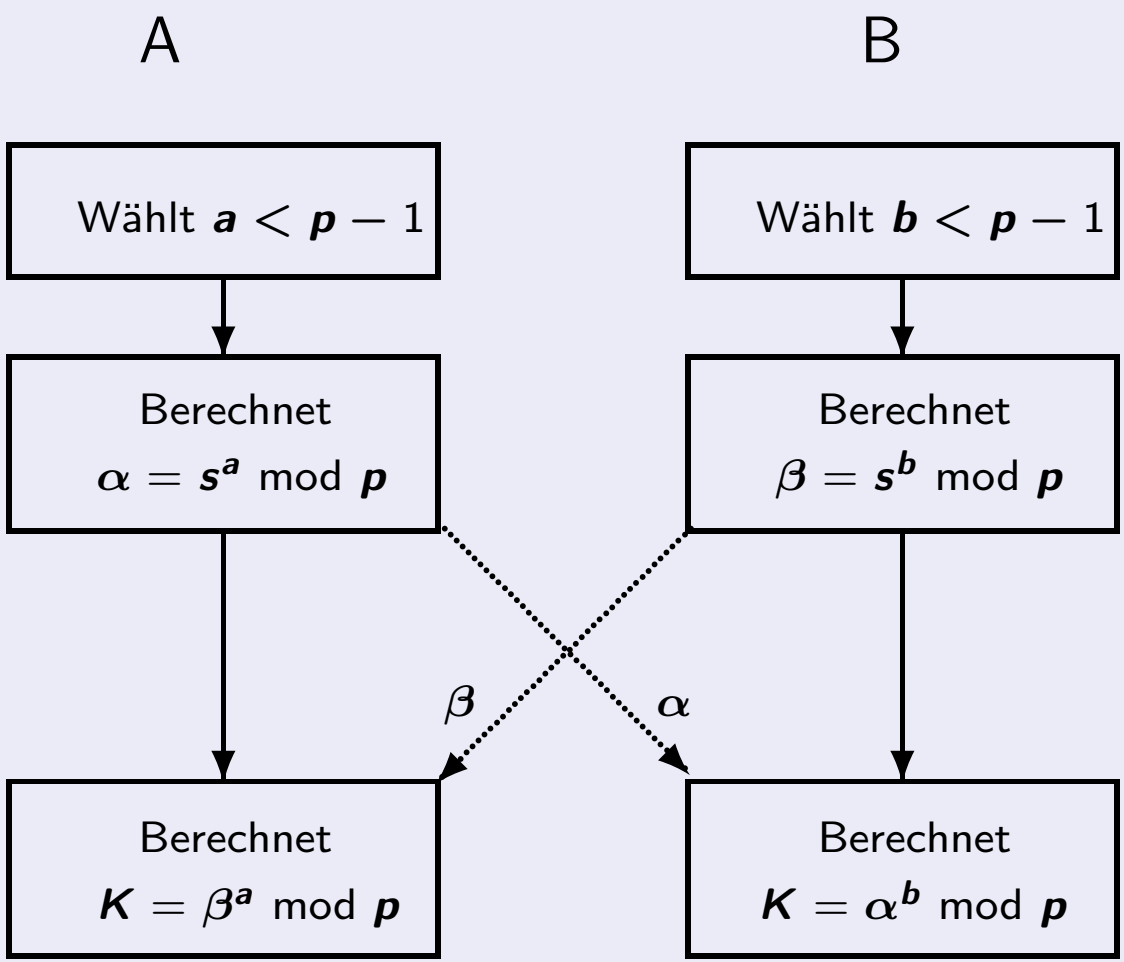
Diffie-Hellman Schlüsseltausch

DHKE (Diffie/Hellman Key Exchange). W. Diffie, M. Hellman (1976)

Gegeben:

Primzahl p,

Zahl $s < p - 1, s \in N$



Diskreter Logarithmus:

$s^a \bmod p \Rightarrow a = ???$

Diffie-Hellman Schlüsseltausch (Forts.)

Besondere Eigenschaft der Zahl s

- damit prinzipiell alle möglichen Schlüssel K generiert werden können, muss s als Primitivwurzel von p gewählt werden
- diese Eigenschaft garantiert, dass s ein sog. Erzeuger ist,
  d. h. $s^k \bmod p$ mit $k \in \{0, 1, ...p-1\}$ erzeugt die Zahlenfolge
  mit allen Zahlen von 1 bis p – 1 in beliebiger Reihenfolge
- falls s keine Primitivwurzel von p ist, ist es für einen Angreifer leichter, den Schlüssel zu bestimmen

Primitivwurzel

- Sei {PW} die Menge aller Primitivwurzeln von p
- es gibt $\varphi(\varphi(p))$ Primitivwurzeln von p, d. h. $|\{PW\}| = \varphi(\varphi(p))$
  [Buch:2016]
- falls p prim ist, ist daher $|\{PW\}| = \varphi(p-1)$
- Beispiel: $p = 7$
  $|\{PW\}| = \varphi(6) = 2$, denn nur die Zahlen 1 und 5 sind teilerfremd zu 6
  $\Rightarrow$ es gibt 2 Primitivwurzeln von 7 (aber welche?)

Diffie-Hellman Schlüsseltausch (Forts.)

Test, ob eine Zahl s eine Primitivwurzel von p ist

- Variante 1: Brute-Force
  (1) x := 1; n := 1
  (2) berechne $x := x \cdot s \bmod p$; $n := n + 1$
  (3) falls $x \neq 1 \Rightarrow$ zurück zu (2)
  (4) falls $n = p \Rightarrow s$ ist Primitivwurzel von p

  $\Rightarrow$ leider für große Zahlen sehr aufwändig!

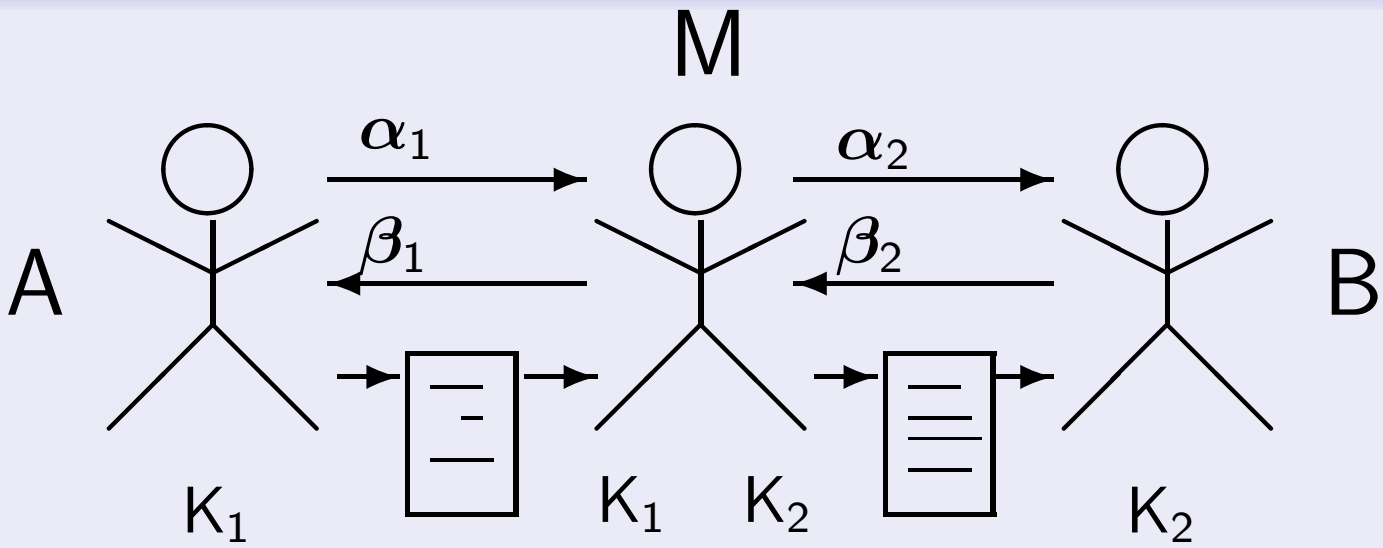
- Variante 2, falls $p = 2q + 1$ mit p, q prim
  - es gilt $|\{PW\}| = \varphi(p-1) = \varphi(2q) = \varphi(q) = q - 1$
  - falls $s^2 \equiv 1 \bmod p$ oder $s^q \equiv 1 \bmod p \Rightarrow s$ ist **keine** Primitivwurzel von p
  - anderenfalls ist s eine Primitivwurzel von p [Ecke:2018]

Beispiel

- Zeigen Sie nach beiden Varianten, dass 3 eine Primitivwurzel von 7 ist
- Zeigen Sie nach beiden Varianten, dass 2 keine Primitivwurzel von 7 ist

Middle-Person Attacke (Man-in-the-Middle Attacke)

Möglicher Angriff



M hat folgende Möglichkeiten:

- Blockieren der Kommunikation
- Abhören der Kommunikation
- Verändern von Nachrichten

Middle-Person Attacke (Forts.)

Abwehrversuch

- Beobachtung:
  - $\boldsymbol{a}$ ist geheim
  - α ist (potenziell) öffentlich bekannt
- Idee:
  α, β, ... in öffentlichem Schlüsselverzeichnis hinterlegen

Weiterhin mögliche Angriffe von M

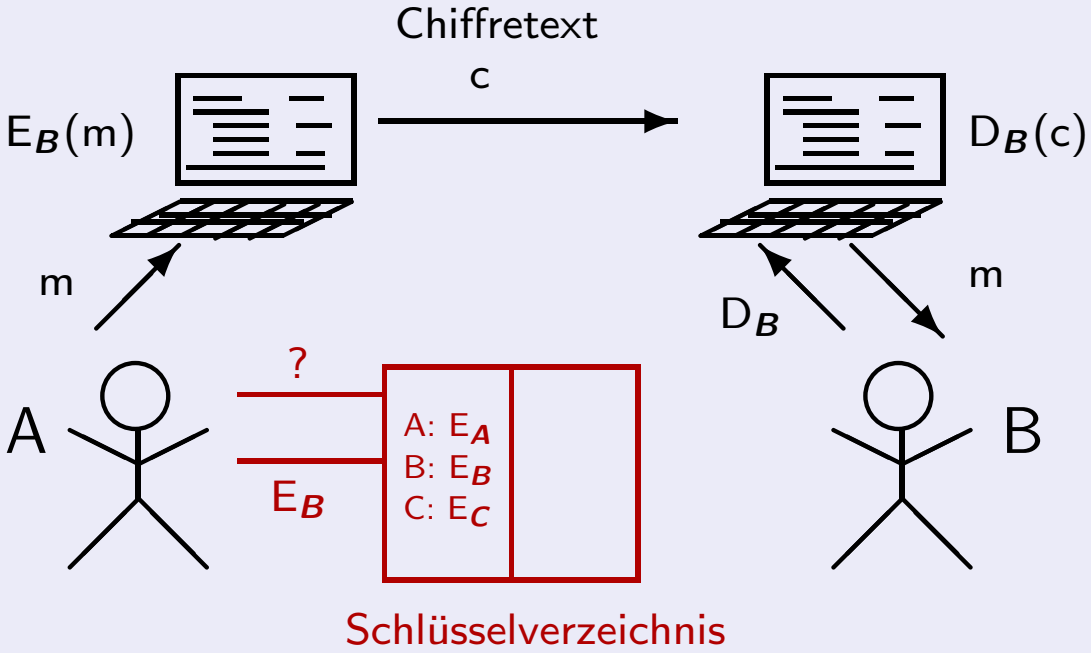
- Veränderung der Verzeichniseinträge des Schlüsselverzeichnisses
- Angriff auf den Kommunikationsweg zum Schlüsselverzeichnis

Was ist DHE?

- *Ephemeral Diffie-Hellman*: die Parameter $\boldsymbol{a}$ und $\boldsymbol{b}$ werden jeweils neu und zufällig gewählt
- dies ermöglicht die sogenannte *Forward Secrecy*

Asymmetrische Verschlüsselung

Prinzip: Schlüsselpaare



Schlüsselpaar von X:

E_X = encryption key von X

D_X = decryption key von X (geheim!)

Wichtige Public-Key Verfahren

- RSA (*zu lösen:* Primzahlprodukt-Faktorisierung)
- ElGamal (*zu lösen:* diskreter Logarithmus)
- Elliptische Kurven (*kurz:* EC)

Das RSA-Verfahren

(RSA = Rivest, Shamir, Adleman)

Prinzip

- $n = p \cdot q$ mit p, q Primzahl
  $z = \varphi(n) = (p-1)(q-1)$
- Wähle e, so dass $1 < e < n$ und
  e relativ prim $z \Rightarrow ggt(z, e) = 1$
- Bestimme d, so dass $(e \cdot d) \equiv 1 \bmod z$ (modulare Inverse zu e in $\mathbb{Z}_z$)

  $\Rightarrow$ resultierendes Schlüsselpaar: $E = (e, n)$ sowie $D = (d, n)$
- Verschlüsseln von m mit E: $m^e \equiv c \bmod n$
- Entschlüsseln von c mit D: $c^d \equiv m \bmod n$

Beobachtungen:

- Zu jedem e existiert genau ein d.
- e und d sind zueinander symmetrisch (und daher austauschbar).

Das RSA-Verfahren (Forts.)

Ein Beispiel

- Gegeben sei die Abbildung zur Kodierung des Klartextes:
  [A:Z] $\longrightarrow$ [2:27]
- Belauschte Chiffretextfolge $\boldsymbol{c}$ von A an B:
  $\boldsymbol{c} = (62; 16; 74; 45; 21)$
- Bekannt: $\boldsymbol{E_B} = (53, 77)$
- **Wie lautet die Klartextfolge $\boldsymbol{m}$?**

Frage: *0 und 1 sind keine brauchbaren Kodierungen. Wieso nicht?*

Das RSA-Verfahren (Forts.)

Vorgehen des Angreifers

- Problem: $\boldsymbol{E} = (\boldsymbol{e}, \boldsymbol{n}) = (53, 77)$ ist zu klein und leicht zerlegbar:
  $\boldsymbol{n} = 77 = 7 \cdot 11 \Rightarrow \boldsymbol{z} = 60$
- Man findet: $(53 \cdot \boldsymbol{d}) \equiv 1 \bmod 60 \Rightarrow \boldsymbol{d} = 17$ (z. B. durch Probieren)
  $\Rightarrow \boldsymbol{D} = (17, 77)$
- Entschlüsselung der Chiffretextfolge mit $\boldsymbol{D} \Rightarrow \boldsymbol{m} \equiv \boldsymbol{c}^{\boldsymbol{d}} \bmod \boldsymbol{n}$

verschlüsselt	entschlüsselt	Klartextzeichen
62	6	E
16	25	X
74	2	A
45	12	K
21	21	T

Frage: *Wieso sollte man nicht Buchstabe für Buchstabe verschlüsseln?*

Das RSA-Verfahren (Forts.)

Über p, q und z

- Wenn z bekannt wird, kann die Zahl n leicht in p und q zerlegt werden.
- Beispiel: $E = (3\,553, 259\,313)$ und $z = 258\,048$
- Man findet:

$$
\begin{aligned}
p \cdot q &= n \\
(p-1) \cdot (q-1) &= p \cdot q - (p+q) + 1 = z \\
\Rightarrow \quad & p + q = n - z + 1 \\
\text{Sei} \quad & A := p + q \qquad (1) \\
(p+q)^2 &= p^2 + 2 \cdot p \cdot q + q^2 = A^2 \\
(p-q)^2 &= p^2 - 2 \cdot p \cdot q + q^2 = A^2 - 4 \cdot n \\
\Rightarrow p - q &= \sqrt{A^2 - 4 \cdot n} \qquad (2)
\end{aligned}
$$

$$
\begin{aligned}
\text{aus (1) + (2)} \Rightarrow p &= \frac{1}{2}\left(A + \sqrt{A^2 - 4 \cdot n}\right) \\
\text{aus (1) - (2)} \Rightarrow q &= \frac{1}{2}\left(A - \sqrt{A^2 - 4 \cdot n}\right) \\
\text{mit} \quad A &= n - z + 1.
\end{aligned}
$$

$\Rightarrow$ A = 1265; $p = 1009$ und $q = 257$

Das RSA-Verfahren (Forts.)

Euklidischer Algorithmus

- berechnet den ggt zweier ganzer Zahlen:
  $ggt(r_0, r_1) = ggt(r_1, r_0 \bmod r_1)$
- durch rekursive Anwendung erhält man
  $ggt(r_0, r_1) = ggt(x, 0) \Rightarrow x$ ist der gesuchte Wert
- Beispiel:
  $ggt(973, 301) = ggt(301, 70) = ggt(70, 21) = ggt(21, 7) = ggt(7, 0)$

Erweiterter Euklidischer Algorithmus

- berechnet neben dem ggt eine Zerlegung, so dass dieser als Linearkombination der Ausgangswerte dargestellt wird:
  $ggt(r_0, r_1) = s \cdot r_0 + t \cdot r_1$
- wichtige Anwendung: Berechnung der modularen Inversen
  (Erinnerung: diese existiert nur dann, wenn $ggt(r_0, r_1) = 1$)
- Beispiel: $r_0 = 258048 = z$; $r_1 = 3553 = e$ (s. nächste Folie)

Das RSA-Verfahren (Forts.)

Berechnen von d mit dem Erweiterten Euklidischen Algorithmus

$$
\begin{aligned}
258\,048 &= 72 \cdot 3\,553 + 2\,232 \\
3\,553 &= 1 \cdot 2\,232 + 1\,321 \\
2\,232 &= 1 \cdot 1\,321 + 911 \\
1\,321 &= 1 \cdot 911 + 410 \\
911 &= 2 \cdot 410 + 91 \\
410 &= 4 \cdot 91 + 46 \\
91 &= 1 \cdot 46 + 45 \\
46 &= 1 \cdot 45 + 1
\end{aligned}
$$

Wir erhalten die folgenden Zerlegungen in Linearkombinationen:

$$
\begin{aligned}
1 &= 46 - 1 \cdot 45 \\
&= 46 - 1 \cdot (91 - 1 \cdot 46) && = 2 \cdot 46 - 1 \cdot 91 \\
&= 2 \cdot (410 - 4 \cdot 91) - 1 \cdot 91 && = 2 \cdot 410 - 9 \cdot 91 \\
&= 2 \cdot 410 - 9 \cdot (911 - 2 \cdot 410) && = 20 \cdot 410 - 9 \cdot 911 \\
&= 20 \cdot (1\,321 - 1 \cdot 911) - 9 \cdot 911 && = 20 \cdot 1\,321 - 29 \cdot 911 \\
&= 20 \cdot 1\,321 - 29 \cdot (2\,232 - 1 \cdot 1\,321) && = 49 \cdot 1\,321 - 29 \cdot 2\,232 \\
&= 49 \cdot (3\,553 - 1 \cdot 2\,232) - 29 \cdot 2\,232 && = 49 \cdot 3\,553 - 78 \cdot 2\,232 \\
&= 49 \cdot 3\,553 - 78 \cdot (258\,048 - 72 \cdot 3\,553) && = \underline{5\,665} \cdot 3\,553 - 78 \cdot 258\,048
\end{aligned}
$$

$\Rightarrow \boldsymbol{D} = (5\,665, 259\,313)$

Das RSA-Verfahren: Rechentechnik

Schema für den Erweiterten Euklidischen Algorithmus

z	mod e	$= r_0$	z	div e	$= x_0$
e	mod r_0	$= r_1$	e	div r_0	$= x_1$
r_0	mod r_1	$= r_2$	r_0	div r_1	$= x_2$
$\cdots$			$\cdots$		
r_{t-2}	mod r_{t-1}	$= 1$	r_{t-2}	div r_{t-1}	$= x_t$

Nun gilt folgende Rekursionsformel:

$$A_k = A_{k+2} - x_k \cdot A_{k+1}; \qquad k = t, \ldots, 0$$

mit der Verankerung $A_{t+2} = 0; \quad A_{t+1} = 1$

Daraus resultiert der gesuchte Wert d mit $d = A_0$

$\Rightarrow$ Berechnen Sie zur Übung: $z = 60, e = 53$

Das RSA-Verfahren: Rechentechnik (Forts.)

Berechnen von m^e mod n

- Problem: wir rechnen mit extrem großen Zahlen
  (i. Allg. $\geq$ 600 Dezimalstellen)!
- Lösung: Ägyptisches Potenzieren (auch: *Square-and-Multiply*)
- Schema:

Exponent	Faktor	relevant?
$e_1 = e$	$m_1 = m$	falls e_1 ungerade
$e_2 = e_1$ **div** 2	$m_2 = (m_1)^2$ mod n	falls e_2 ungerade
$e_3 = e_2$ **div** 2	$m_3 = (m_2)^2$ mod n	falls e_3 ungerade
$\ldots$	$\ldots$	
1	$m_\ell = (m_{\ell-1})^2$ mod n	ja

Ergebnis durch Aufmultiplizieren nur der relevanten Faktoren m_k ($\Rightarrow T_i$):

$$E(m) = \prod_i T_i \bmod n$$

- Vorteil: logarithmische Komplexität der Berechnung

Das RSA-Verfahren: Rechentechnik (Forts.)

Beispiel:

- Aufgabe: Berechnen von $62^{17} \bmod 77$
- Lösung:

Exponent	Faktor	relevant?
17	62	• (T_1)
8	-6	
4	36	
2	-13	
1	15	• (T_2)

$$\Rightarrow 62^{17} \bmod 77 = 62 \cdot 15 \bmod 77 = 6$$

Primzahlen

Primzahlen

- p und q müssen große Primzahlen (ca. $n/2$) sein und zufällig gewählt werden

  p und q sollten annähernd die gleiche Länge besitzen
- es gibt unendlich viele Primzahlen (Beweis durch Euklid)
- aber sie werden seltener, je größer die Zahlen werden
- für eine zufällig gewählte ungerade Zahl x gilt:

$$\textbf{Prob}(x \text{ ist prim}) \approx \frac{2}{\ln(x)}$$

- nach der Wahl muss x unbedingt auf die Prim-Eigenschaft getestet werden; falls x nicht prim ist, muss ein neues x zufällig gewählt werden

  $\Rightarrow$ eine systematische Konstruktion von Primzahlen könnte nachvollzogen werden und ist daher potenziell unsicher
- starke Primzahlen
  - liegen nicht zu dicht beieinander
  - die Zahlen $x - 1$ und $x + 1$ müssen große Primfaktoren besitzen

Primzahlen (Forts.)

Faktorisierung

- Zerlegen einer Zahl x in ihre Primfaktoren
  Beispiel: $208\,568 = 2^3 \cdot 29^2 \cdot 31$

Faktorisierung: Teilermethode

- Teste alle (Prim-)Zahlen kleiner oder gleich $\sqrt{x}$, ob sie x ohne Rest teilen
- für große x müssen alle ungerade Zahlen getestet werden, da für große Zahlen i. Allg. nicht bekannt ist, ob sie prim sind
  $\Rightarrow$ zu aufwändig bei großen x!

Einige weitere Faktorisierungsmethoden

- GNFS (General Number Field Sieve)
  schnellste bekannte Methode für Zahlen größer als 10^{110}
- Quadratisches Zahlenfeldsieb
  schnellste bekannte Methode für Zahlen kleiner als 10^{110}
- Pollards Rho Algorithmus
  probabilistische Methode

Primzahlen (Forts.)

Faktorisierung: Idee des Quadratischen Zahlenfeldsiebs

- Gesucht: Teiler von x
- Finde zwei natürliche Zahlen a und b, für die gilt:

  $a^2 \equiv b^2 \bmod x$

  $a^2 - b^2 \equiv 0 \bmod x$

  $(a + b)(a - b) \equiv 0 \bmod x$

  also sind $(a + b)$ und $(a - b)$ Teiler von x (oder Vielfache eines Teilers)
- ein einfaches Beispiel:
  $x = 35$

  $35 + 1$ ist eine Quadratzahl $\Rightarrow$ wähle $a = 6$ und $b = 1$

  es gilt: $6^2 - 1^2 = 35 \equiv 0 \bmod 35$

  $\Rightarrow 6 - 1 = 5$ sowie $6 + 1 = 7$ sind Teiler von 35
- weitere Beispiele zum Ausprobieren: $x = 45$, $x = 407$

Primzahlen (Forts.)

Primzahltests

- eine Tabelle mit allen Primzahlen zu erstellen, ist nicht praktikabel
  - Primzahlsatz: es gibt ca. $\frac{x}{ln(x)}$ Primzahlen kleiner als x
    $\Rightarrow$ für $x = 2^{2048}$ gibt es mehr als 10^{600} Primzahlen!
- ein Test, ob eine Zahl prim ist, ist viel effizienter als die Faktorisierung
- deterministische Tests
  - existieren (z. B. AKS-Primzahltest), aber mit schlechtem Laufzeitverhalten
- meistens werden probabilistische Tests eingesetzt

Probabilistischer Test nach dem kleinen Fermatschen Satz

- kann *nicht sicher* entscheiden, ob die getestete Zahl x prim ist
  - IN: x (zufällig gewählte ungerade Zahl) und s (Sicherheitsparameter)
    (1) wähle eine zufällige Zahl $a \in \{2, 3, \ldots, x-2\}$
    (2) falls $a^{x-1} \not\equiv 1 \bmod x$ $\Rightarrow$ OUT: x ist nicht prim, ENDE
    (3) durchlaufe Schleife zu (1) s-mal
    OUT: x ist wahrscheinlich prim, ENDE

Primzahlen (Forts.)

Probabilistischer Test nach Miller-Rabin

(vgl. [PP2016], S. 218)

- höhere Sicherheit als der Test nach Fermat
  - IN: x (zu testende Zahl) und
    a (zufällig gewählte ungerade Zahl mit $a \in \{1, ..., x-1\}$)
  - OUT: x ist wahrscheinlich prim oder x ist nicht prim
- Laufzeitverhalten $O(\log \frac{1}{\epsilon} \cdot (\log x)^3)$ (ϵ ist die Fehlerwahrscheinlichkeit)
  $\Rightarrow$ ca. 100 sec. für 100-stellige Dezimalzahl bei $\epsilon = 10^{-100}$ und 10^6 Op/sec

Praktische Implementierung von Primzahltests

- häufig gewähltes Vorgehen
  (1) erzeuge einen zufälligen Bit-String x der Länge n
  (2) setze des erste und das letze Bit von x auf 1
  (3) prüfe, dass x keine Teiler kleiner als 256 besitzt
  (4) durchlaufe den Miller-Rabin-Test mit $s \geq 5$ unterschiedlichen, zufällig gewählten Zahlen a (Fehlerwahrscheinlichkeit $\epsilon \leq (\frac{1}{4})^s$)
  (5) falls alle Tests erfolgreich waren, ist x (vermutlich) eine Primzahl

Sicherheit von RSA

Schlüssellängen

- Im Jahr 2005 wurde RSA-664 faktorisiert
  (Uni Bonn. 170 Pentium-1-GHz-Jahre, GNFS).
- Im Dezember 2009 wurde RSA-768 faktorisiert
  (Lausanne, Tokio, Bonn, Nancy, Redmond, Amsterdam.
  2.5 Jahre; mehrere hundert Rechner, ca. 1500 AMD64-Jahre)
- Im Jahr 2019 wurde RSA-795 faktorisiert
  (Internationales Team, 900 CPU-Kern-Jahre auf 2.1 GHz Xeon Gold)
- äquivalente Sicherheitsniveaus (nach heutigem Wissen):

  1024 Bit (RSA) $\widehat{=}$ 80 Bit (symm.)
  2048 Bit (RSA) $\widehat{=}$ 112 Bit (symm.)
  3072 Bit (RSA) $\widehat{=}$ 128 Bit (symm.)
  7680 Bit (RSA) $\widehat{=}$ 192 Bit (symm.)
  15360 Bit (RSA) $\widehat{=}$ 256 Bit (symm.)

- seit 2014 empfohlene Mindestschlüssellänge (nach BSI): 2048 Bit

Sicherheit von RSA (Forts.)

Eine gute Idee?

- Wir erstellen eine Lookup-Tabelle mit vorberechneten Faktorisierungen für alle möglichen Zahlen $n = p \cdot q$
- Beobachtung: das Erstellen dauert SEHR lange ...
- Wie groß wird die Tabelle?

  Grobe Abschätzung für RSA-665:

  - es gibt 10^{200} Zahlen der Bitlänge 665
  - um $10^{200} \cdot 665$ Bit zu speichern, brauchen wir ca. 10^{190} Festplatten der Größe 1 TB
  - wenn jede Festplatte nur 10^{-6} Gramm (!) leicht wäre, wären es $7.7 \cdot 10^{178}$ Tonnen

  $\Rightarrow$ Problem: Wir haben ein schwarzes Loch!!!

Sicherheit von RSA (Forts.)

Geschwindigkeit

- für eine Verschlüsselung mit RSA-2048 benötigt man mehrere Millionen Integer-Multiplikationen (ca. 10 ms pro Verschlüsselung)

  $\Rightarrow$ RSA nicht für große Datenmengen einsetzen!
  (Verschlüsselungsrate ca. 2 Mbps, AES ist ungefähr 1000 mal schneller!)

Sicherheit

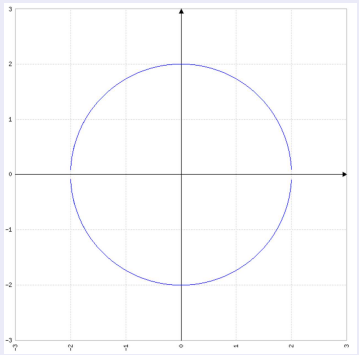
- $\boldsymbol{p}$ und $\boldsymbol{q}$ sollten starke Primzahlen von ausreichender Größe sein
- keine zwei Teilnehmer sollten dasselbe $\boldsymbol{n}$ besitzen
- $\boldsymbol{e}$ und $\boldsymbol{d}$ dürfen nicht zu klein gewählt werden ($\boldsymbol{d} > \frac{1}{3}\boldsymbol{n}^{\frac{1}{4}}$; Satz von Wiener)
- Nachrichtenblöcke $\boldsymbol{m}$ sollten weniger Stellen als $\boldsymbol{n}$ haben und mit zufälligen Bits aufgefüllt werden (vgl. *Optimal Asymmetric Encryption Padding*)
- falls ein implementierbarer, effizienter Faktorisierungsalgorithmus gefunden werden sollte, ist RSA gebrochen
- falls ein funktionierender Quantencomputer entwickelt werden sollte, ist RSA gebrochen

Elliptische Kurven

Idee

- Kreisgleichung (Kreis mit Radius r):

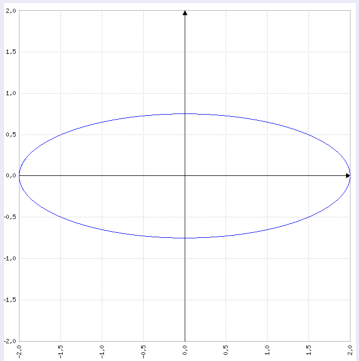
$$x^2 + y^2 = r^2, \quad \text{mit } x, y, r \in \mathbb{R}$$



$x^2 + y^2 = 4$

- Ellipsengleichung (Breite$= \sqrt{\frac{1}{a}}$; Höhe$= \sqrt{\frac{1}{b}}$);

$$ax^2 + by^2 = 1, \quad \text{mit } x, y, a, b, c \in \mathbb{R}$$



$\frac{1}{4}x^2 + \frac{16}{9}y^2 = 1$

Elliptische Kurven (Forts.)

Elliptische Kurve

- die Punkte einer *elliptische Kurve* werden über einem Primkörper $\mathbb{Z}_p$ definiert:

$$y^2 \equiv x^3 + a \cdot x + b \bmod p \quad \text{mit } x, y, a, b \in \mathbb{Z}_p \text{ und } 4 \cdot a^3 + 27 \cdot b^2 \not\equiv 0 \bmod p$$

- nur diejenigen Punkte $P(x, y)$, welche diese Gleichung erfüllen, gehören zur elliptischen Kurve
- zusätzlich wird ein Punkt $\mathcal{O}$ als *neutrales Element* definiert:

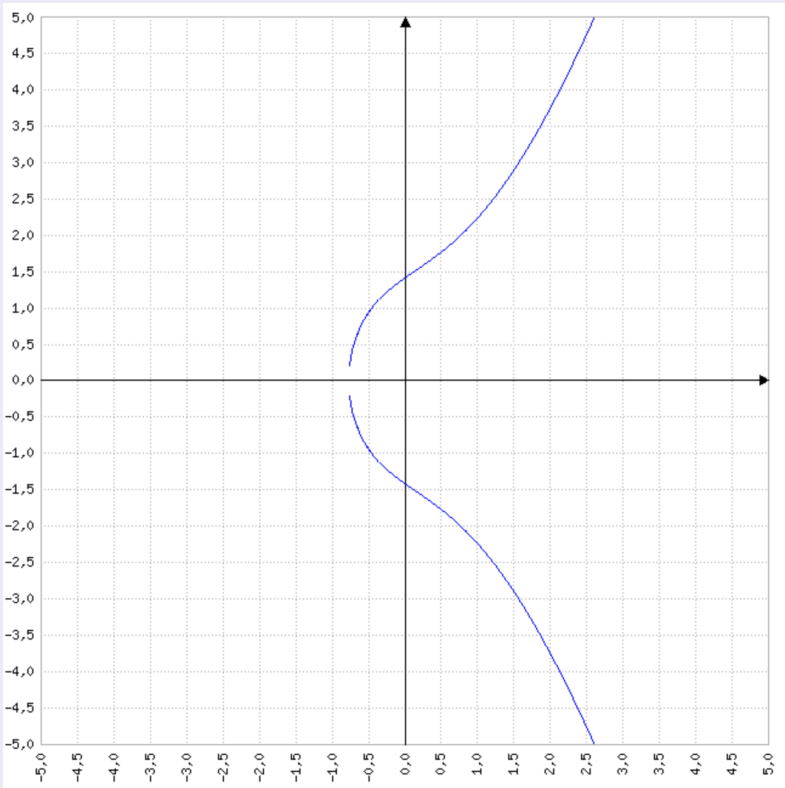
$$P + \mathcal{O} = P$$

Elliptische Kurven (Forts.)

Beispiel: $a = 2, b = 2, p = 17$

$\Rightarrow$ elliptische Kurvengleichung: $y^2 \equiv x^3 + 2 \cdot x + 2 \bmod 17$

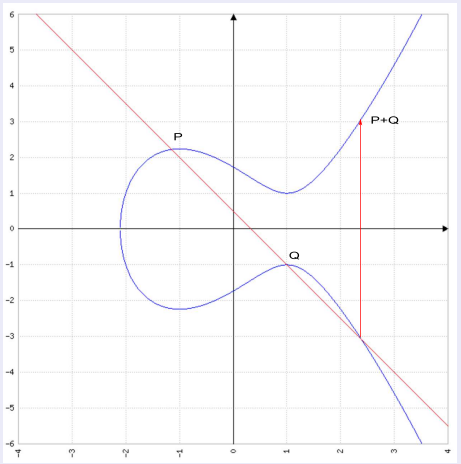
- eine grafische Darstellung in $\mathbb{Z}_p$ ist nicht hilfreich
- grafische Darstellung von $y^2 = x^3 + 2 \cdot x + 2$ über $\mathbb{R}$



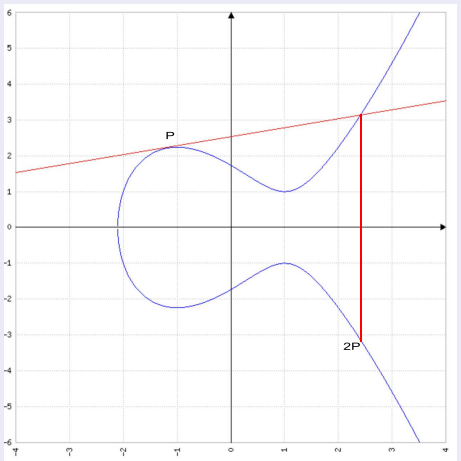
Elliptische Kurven (Forts.)

Definition der Rechenoperationen

- **Punktaddition**



- **Punktverdoppelung**



Elliptische Kurven (Forts.)

Rechenoperationen auf elliptischen Kurven [PP2016]

- **Punktaddition**: $Z(x_3, y_3) = P(x_1, y_1) + Q(x_2, y_2), \quad P, Q \neq \mathcal{O}$

  falls $P \neq Q$:

  falls $x_1 = x_2 \Rightarrow Z = \mathcal{O}$

  falls $x_1 \neq x_2$:

$$\begin{aligned} s &\equiv (y_2 - y_1) \cdot (x_2 - x_1)^{-1} \bmod p \\ x_3 &\equiv s^2 - x_1 - x_2 \bmod p \\ y_3 &\equiv s \cdot (x_1 - x_3) - y_1 \bmod p \end{aligned}$$

  falls $P = Q$: $\quad (\Rightarrow$ **Punktverdoppelung** $Z = 2P)$

  falls $y_1 = 0 \Rightarrow Z = \mathcal{O}$

  falls $y_1 \neq 0$:

$$\begin{aligned} s &\equiv (3 \cdot x_1^2 + a) \cdot (2 \cdot y_1)^{-1} \bmod p \\ x_3 &\equiv s^2 - 2 \cdot x_1 \bmod p \\ y_3 &\equiv s \cdot (x_1 - x_3) - y_1 \bmod p \end{aligned}$$

Elliptische Kurven (Forts.)

Beispiel

- Gegeben $\boldsymbol{P} = (5, 1)$ mit $\boldsymbol{y}^2 \equiv \boldsymbol{x}^3 + 2 \cdot \boldsymbol{x} + 2 \bmod 17$
- Rechnung für $\boldsymbol{Z} = 2\boldsymbol{P}$:

  $\Rightarrow \boldsymbol{s} \equiv (3 \cdot 5^2 + 2) \cdot (2 \cdot 1)^{-1} \bmod 17 = 77 \cdot 2^{-1} \equiv 9 \cdot 9 \equiv 13 \bmod 17$

  $\boldsymbol{x}_3 \equiv 13^2 - 2 \cdot 5 \bmod 17 = 159 \equiv 6 \bmod 17$

  $\boldsymbol{y}_3 \equiv 13 \cdot (5 - 6) - 1 \bmod 17 = -14 \equiv 3 \bmod 17$

  $\Rightarrow 2\boldsymbol{P} = (6, 3)$
- Rechnung für $\boldsymbol{Z} = \boldsymbol{P} + 2\boldsymbol{P} = 3\boldsymbol{P}$:

  $\Rightarrow \boldsymbol{s} \equiv (3 - 1) \cdot (6 - 5)^{-1} \bmod 17 = 2 \cdot 1^{-1} \equiv 2 \cdot 1 \equiv 2 \bmod 17$

  $\boldsymbol{x}_3 \equiv 2^2 - 5 - 6 \bmod 17 = -7 \equiv 10 \bmod 17$

  $\boldsymbol{y}_3 \equiv 2 \cdot (5 - 10) - 1 \bmod 17 = -11 \equiv 6 \bmod 17$

  $\Rightarrow 3\boldsymbol{P} = (10, 6)$
- Berechnen Sie $4\boldsymbol{P}$ mittels $2(2\boldsymbol{P})$ und $3\boldsymbol{P} + \boldsymbol{P}$ sowie $5\boldsymbol{P}, \ldots, 21\boldsymbol{P}$

DHKE mit elliptischen Kurven (ECDH)

ECDH-Domain-Parameter

- Primzahl p, a, b sowie ein primitives Element $PE = (x, y)$ bilden die *Domain-Parameter* zur Kurve $y^2 = x^3 + ax + b \bmod p$
- Vorsicht: nicht alle Kurven sind kryptografisch stark!

ECDH-Schlüsseltausch

Gegeben: Domain-Parameter der EC, n = Anzahl der Gruppenelemente

- Alice:
  (1) wähle ein zufälliges $d_A \in \{2, 3, \ldots, n-1\}$ (ausreichend groß)
  (2) berechne $PA = d_A \cdot PE$
  (3) sende PA an Bob und empfange PB von Bob
  (4) berechne $T_{AB} = d_A \cdot PB = (x_{AB}, y_{AB})$ als gemeinsames Geheimnis
  (5) gemeinsamer Schlüssel $K_{AB} = x_{AB}$ (oder y_{AB})
- Bob:
  - analog
- n kann näherungsweise mit $p + 1 - 2\sqrt{p}$ abgeschätzt werden [PP2016]

DHKE mit elliptischen Kurven (ECDH) (Forts.)

Sicherheit

- Angreifer muss den diskreten Logarithmus

$$d_a = log_p(PA)$$

oder

$$d_b = log_p(PB)$$

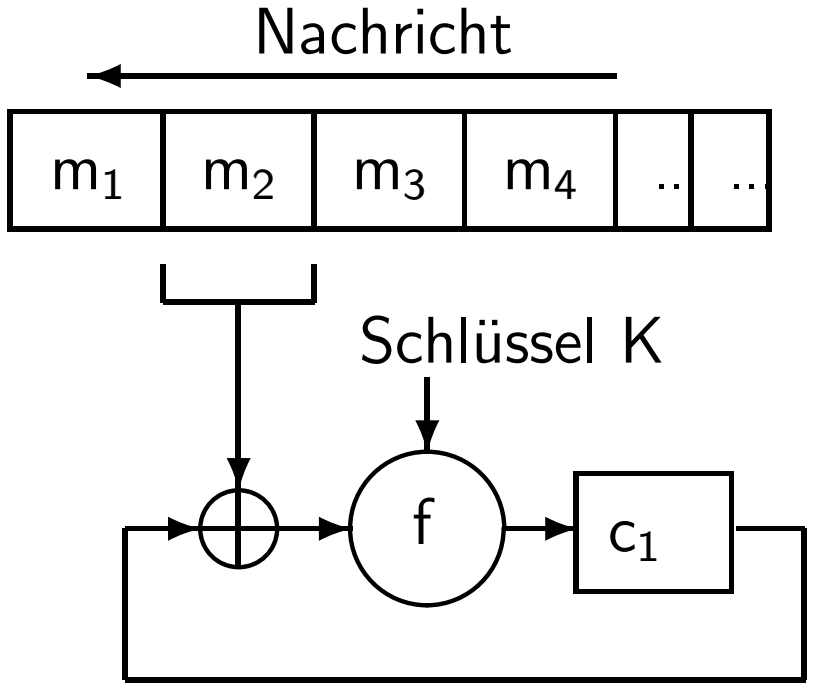
berechnen

- aufgrund der aufwändigen Berechnungen auf einer elliptischen Kurve kann mit kleineren Zahlen hantiert werden als bei DHKE
- eine Wahl der Primzahl p mit 256 **Bit** bietet ein derzeit ausreichendes Sicherheitsniveau von 128 Bit
- kürzere Schlüssel erlauben Implementierungen auf schwächerer Hardware
- Aber: es müssen kryptografisch starke elliptische Kurven gewählt werden!
- die zertifizierten Kurven des NIST sind durch die NSA möglicherweise manipuliert und absichtlich geschwächt (Bernstein und Schneier, 2013)
- es gibt gute Kurven akademischen Ursprungs (Ed25519, Ed448-Goldilocks, E-521)

Gegenseitige Authentisierung

Message Authentication Code (MAC)

- Problem: Von wem stammt eine Nachricht?
- Sei $f(m, K)$ eine geeignete Funktion, die aus einer beliebig langen Nachricht und einem Schlüssel K einen Wert fester Länge generiert ($\Rightarrow$ Prüfsumme) (z.B. CBC-MAC-TDES beim System „Geldkarte“)



Gegenseitige Authentisierung (Forts.)

Authentisierungsvorgang

- Idee: A und B besitzen ein gemeinsames Geheimnis $\boldsymbol{K}$
- jede Nachricht $\boldsymbol{m}$ von A bzw. B wird mit dem Wert $\boldsymbol{MAC} = \boldsymbol{f}(\boldsymbol{m}, \boldsymbol{K})$ „unterzeichnet“, indem dieser Wert neben $\boldsymbol{m}$ mitgesendet wird
- falls $\boldsymbol{f}()$ hinreichend sicher ist, muss die Nachricht mit dem geheimen Schlüssel $\boldsymbol{K}$ unterzeichnet worden sein
- A kann verifizieren, dass eine empfangene Nachricht von B stammt (und umgekehrt)
- Nachricht $\boldsymbol{m}$ muss selbst nicht unbedingt verschlüsselt werden

Handschriftliche Unterschrift

Eigenschaften einer Unterschrift unter einem Dokument

- bewusst erstellt $\Rightarrow$ bewusste Willenserklärung
- fälschungssicher $\Rightarrow$ authentisch
- nicht wiederverwendbar $\Rightarrow$ nicht einfach in ein anderes Dokument kopierbar
- unveränderbar $\Rightarrow$ sichert die Integrität des Dokuments
- nicht abstreitbar $\Rightarrow$ Beweisbarkeit gegenüber Dritten

Frage: *Wie lässt sich dies in die digitale Welt übertragen?*

Digitale Signatur

Begriffe

- die Begriffe *Digitale Signatur* und *Elektronische Signatur* (bzw. *Unterschrift*) werden i. allg. Sprachgebrauch synonym verwendet
- der Begriff *Qualifizierte Digitale Signatur* stammt aus dem deutschen Signaturgesetz (SigG)
  - eine qualifizierte Signatur setzt u. a. eine sichere Signaturerstellungseinheit (z. B. eine Smartcard) voraus
  - zur handschriftlichen Unterschrift als gleichwertig anerkannt
- seit 2016 ist die europäische eIDAS-Verordnung maßgebend
  - auch cloudbasierte Signaturen können als qualifizierte Signaturen gelten
  - vollständige Online-Identifizierung anerkannt

Ziele

- Nachrichtenauthentizität: Wer ist der Verfasser?
- Beweisbarkeit der Nachrichtenintegrität
- Nichtabstreitbarkeit der Autorenschaft (*Non-Repudiation*)

Digitale Signatur (Forts.)

Idee

- Verwende ein asymmetrisches Verfahren (z. B. RSA) jedoch in seiner Umkehrung!

RSA-Signatur

(1) RSA-Schlüsselpaar erzeugen

(2) (kurze) Nachricht m signieren mittels $s \equiv m^d \bmod n$

(3) sende (m, s)

Verifikation einer RSA-Signatur

(1) empfange (m', s')

(2) öffentlichen Schlüssel des Senders ausfindig machen

(4) falls $m' \equiv (s')^e \bmod n$
$\Rightarrow s'$ wurde mit dem privaten Schlüssel des Senders erzeugt
$\Rightarrow m'$ stammt vom Sender

Digitale Signatur (Forts.)

Angriff: existenzielle Fälschung

(1) M wählt eine Zahl $s \in \{0, \ldots, n-1\}$
M wiederholt den Schritt (1) solange bis $m \equiv s^e \bmod n$ eine sinnvolle Nachricht ist

(2) M behauptet, s sei eine Signatur von A

(3) die Verifikation ergibt die Nachricht m, die scheinbar von A stammt

Abwehr

- durch Vorschreiben einer speziellen Form wird der Angriff nahezu unmöglich
  Beispiele:
  - Duplizieren: m muss zum Signieren zweimal identisch nacheinander geschrieben werden
  - Verwendung von Padding: m muss eine bestimmte Struktur aufweisen (z. B. 100 Null-Bits am Ende)
- aus Sicherheitsgründen (und Geschwindigkeitsgründen) sollten nicht längere Dokumente signiert werden, sondern eine eindeutige Prüfsumme
  ($\Rightarrow$ Berechnung mittels einer *kryptografischen Hashfunktion*)

Einschub: Kryptografische Hashfunktionen

Einfache Hashfunktionen

- Wie funktioniert Hashing?
- Wozu werden einfache Hashfunktionen eingesetzt?

Definition einer kryptografisch sicheren Hashfunktion H

- IN: beliebig lange Nachricht m
- OUT: Hashwert $h = H(m)$ fester Länge (auch: HMAC)
  $\Rightarrow$ Es muss *Kollisionen* geben!
  $\Rightarrow$ Sicherheit von H hängt maßgeblich von der Hashwertlänge ab

  $\Rightarrow$ bei Änderung eines einzigen Bits in m sollten sich ungefähr die Hälfte der Bits in h in einer nicht vorhersagbaren Weise ändern
- Anforderungen an H:
  - Urbildresistenz
  - Kollisionsresistenz
  - H muss eine deterministische Funktion sein $\Rightarrow$ Wieso?

Einschub: Kryptografische Hashfunktionen (Forts.)

Urbildresistenz

- $\boldsymbol{H}(\boldsymbol{m})$ muss einfach (also schnell) zu berechnen sein
- es muss praktisch unmöglich sein, zu einem gegebenen Hashwert $\boldsymbol{h}$ die ursprüngliche Nachricht $\boldsymbol{m}$ zu berechnen (Umkehrung)
  $\Rightarrow$ $\boldsymbol{H}$ ist eine Einwegfunktion
- Digitale Signaturen sind leicht entschlüsselbare Hashwerte
  $\Rightarrow$ es könnten stark verschlüsselte Nachrichten mit Hilfe ihrer Signatur ggf. leichter entschlüsselt werden
- darüberhinaus muss es praktisch unmöglich sein, zu einem gegebenen Hashwert $\boldsymbol{h}$ irgendeine Nachricht $\boldsymbol{m}$ zu berechnen, die diesen Hashwert besitzt! (*Zweite Urbildresistenz*)

Kollisionsresistenz

- es muss praktisch unmöglich sein, zwei unterschiedliche Nachrichten $\boldsymbol{m}_1$ und $\boldsymbol{m}_2$ zu finden, die denselben Hashwert $\boldsymbol{h}$ besitzen $\Rightarrow$ Kollision
- Kollisionserzeugung ist einfacher als die Berechnung einer Umkehrung ($\Rightarrow$ Geburtstagsparadox)

Einschub: Kryptografische Hashfunktionen (Forts.)

Einsatzgebiete

- Sicherung der Integrität von Dateien, insbesondere auch Systemdateien ($\Rightarrow$ Schutz gegen Malware / Hacker)
- Krypto-Geld
- Speichern von Passwörtern zur Überprüfung
- digitale Signatur

Kryptografische Hashverfahren

- alte, nicht mehr sichere Verfahren (Kollisionserzeugung zu leicht)
  - MD4, MD5 (Hashwertlänge 128 Bit)
  - SHA / SHA-1 (Secure Hash Algorithm, 160 Bit)
- moderne als sicher geltenden Verfahren
  - SHA-2 (seit 2002): SHA-256, SHA-384, SHA-512, SHA-224
  - SHA-3 (*Keccak*-Algorithmus, 224, 256, 384, 512 Bit)
  - Blake2 (256 bzw. 512 Bit)

Einschub: Kryptografische Hashfunktionen (Forts.)

Beispiel: SHA-1

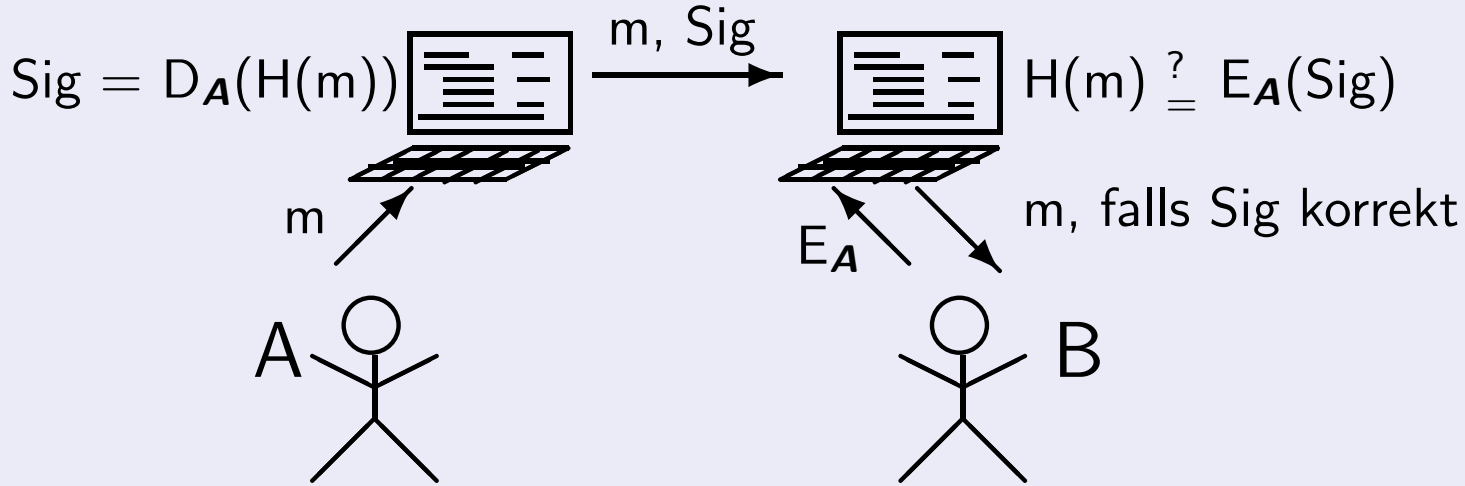
- Secure Hash Algorithm, US-amerikanischer Standard, NIST / NSA 1995
- Merkle-Dåmgard Konstruktion mit Feistel-ähnlichem Aufbau
- der Eingabetext wird 256-Bit Blöcke untergliedert, vorher mittels Padding auf ein Vielfaches von 256 Bit verlängert $\Rightarrow$ jeder Block besteht aus sechzehn 32-Bit Werten (damals Wortlänge in der Hardware)
- Initialisierung mit fünf fest vorgegebenen 32-Bit Werten
- Verarbeitung – unter Einbeziehung der zu hashenden Nachricht – in vier Schritten à 20 Runden zu einem 160-Bit Hashwert (s. [PP2016])
- in jedem Schritt wird eine andere Kompressionsfunktion (zusammengesetzt aus Shift-Operationen, und AND, OR, XOR, NOT Verknüpfungen) verwendet
- schnell in Software
- Hashwertlänge 160 Bit, Sicherheitsniveau 80 Bit,
  aufgrund neuerer Angriffe auf die Kollisionserzeugung vermutlich nur 63 Bit
- SHA-2 baut auf SHA-1 auf: längere Hashwerte, sicherer

Digitale Signatur (Forts.)

Signaturverfahren

- RSA-Signatur
- Elgamal-Signatur
- Merkle-Signatur (resistent gegen Angriffe durch Quantencomputer)
- DSA (Digital Signature Algorithm), mind. 2048 Bit (diskr. Log.)
- ECDSA, 160 – 256 Bit (ellipt. Kurven) $\Rightarrow$ kürzere Signaturen!

Prinzip



Frage: *Woher kennt **B** den öffentlichen Schlüssel von **A**?*

Digitale Signatur (Forts.)

Antwort

- Der Schlüssel wird von einer Stelle, der ***A*** und ***B*** vertrauen, **fälschungssicher** bestätigt.
  $\Rightarrow$ *Trust Center*
  $\Rightarrow$ Zertifizierungsstelle, ***CA*** (*Certification Authority*)
- Bestätigung in Form eines Zertifikates
- ***CA*** darf Zertifikate nicht leichtfertig ausstellen!

Zertifizierungsstelle

Aufgaben einer Zertifizierungsstelle

- Identifizierung und Registrierung
  (gegen Vorlage eines amtlichen Ausweises)
- Schlüsselgenerierung (Schlüsselpaar erstellen)
- Schlüsselzertifizierung (in Form eines Zertifikats)
- Personalisierung (einer Signaturkomponente, z. B. Chipkarte)
- Verzeichnisdienst (zum Abruf von Zertifikaten und Informationen)
- ggf. Zeitstempeldienst
- ggf. Key Recovery

Zertifikat

Welche Informationen enthält ein Zertifikat?

(Minimal-) Aufbau eines Zertifikates	
1.	Name (oder Pseudonym) des Benutzers ggf. mit weiteren Zusätzen
2.	Prüfschlüssel zum Signaturschlüssel
3.	verwendbare Signaturalgorithmen
4.	Seriennummer
5.	Gültigkeitszeitraum
6.	Name der ausstellenden CA
7.	eventuelle Verwendungsbeschränkungen
8.	elektronische Signatur des Zertifikats durch die CA

Frage: *Woher bekomme ich das Zertifikat?*

Zertifikat (Forts.)

Was ist bei der Prüfung (Verifikation) eines Zertifikats zu beachten?

(1) Stimmt der eingetragene Benutzername?

(2) Vertraue ich der ausstellenden CA?

(3) Ist die Signatur der CA korrekt? ($\Rightarrow E_{CA}$)

(4) Ist das Zertifikat abgelaufen?

(5) Wurde das Zertifikat zurückgerufen?
$\Rightarrow$ Certificate Revocation List (CRL)

Frage: *Woher bekomme ich den Schlüssel E_{CA} ?*

Zertifikat (Forts.)

Welche Rolle spielt hier Vertrauen?

- durch die Zertifikate wird eine Vertrauenskette aufgebaut
  $\Rightarrow$ PKI (*Public Key Infrastructure*)
- den Prüfschlüssel $\boldsymbol{E_{CA}}$ erhält man aus dem CA-Zertifikat
- dieses ist von einer übergeordneten CA zertifiziert oder nur selbstsigniert

Technische Richtlinie des BSI

- aktuelle Empfehlungen zu Sicherheitsprotokollen gibt das *Bundesamt für Sicherheit in der Informationstechnik* (BSI) regelmäßig heraus, aktuell:

BSI – Technische Richtlinie
Kryptographische Verfahren: Empfehlungen und Schlüssellängen
BSI TR-02102-1 2019-01
22. Februar 2019

URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf

Identifikation

Challenge-Response-Verfahren

- Alice und Bob haben ein gemeinsames Geheimnis K (z. B. AES-Schlüssel)
- Bob muss Alice beweisen, dass er K kennt
- Bob darf K nicht einfach an Alice senden (Wieso nicht?)
- Protokoll
  (1) Alice wählt eine Zufallszahl R und sendet diese an Bob (Challenge)
  (2) Bob verschlüsselt R mit K und sendet das Ergebnis an Alice (Response)
  (3) Alice überprüft, ob die Entschlüsselung von Bobs Antwort wieder R ergibt (Verifikation)
- Schwächen
  - Mallory kann alle Nachrichten mitlesen und ggf. wiederverwenden
    Wird das Verfahren wiederholt, darf R niemals wiederverwendet werden!
  - ein gemeinsames Geheimnis muss bereits für alle Teilnehmerpaare bestehen

Identifikation (Forts.)

Zero-Knowledge-Beweis nach Fiat-Shamir

- Bob wählt zwei große Primzahlen p und q und berechnet $n = p \cdot q$
- Bob wählt s relativ prim zu n und berechnet $v \equiv s^2 \bmod n$
  $\Rightarrow (v, n)$ dient als öffentlicher und (s, n) als privater Schlüssel
- um Alice zu beweisen, dass er s kennt, ohne s zu nennen, wählt Bob r relativ prim zu n , berechnet $x \equiv r^2 \bmod n$ und sendet x an Alice
- Alice kennt (v, n) und erzeugt eine Zufallszahl $e \in \{0, 1\}$ (Münzwurf) und sendet e an Bob
- falls $e = 0$
  - Bob sendet r an Alice
  - Alice verifiziert, dass $r^2 \equiv x \bmod n$
- falls $e = 1$
  - Bob berechnet $y \equiv r \cdot s \bmod n$ und sendet y an Alice
  - Alice verifiziert, dass $y^2 \equiv x \cdot v \bmod n$
- Alice wiederholt das Verfahren mehrfach, um mit hoher Sicherheit eine falsche Identität von Bob auszuschließen

Identifikation (Forts.)

Zero-Knowledge-Beweis nach Fiat-Shamir (Forts.)

- nur Bob kennt die Zerlegung von n in p und q
- derjenige, der die Quadratwurzel von v berechnen kann, kann auch n faktorisieren (und umgekehrt) (Beweis in [Buch:2016])
  $\Rightarrow$ nur Bob kennt s
- Angriff von Mallory (gibt sich als Bob aus)
  (1) wählt r', berechnet $x' \equiv r'^2 \bmod n$ und sendet x' an Alice
    - falls $e = 0$ (mit Wahrscheinlichkeit 1/2)
      $\Rightarrow$ Mallory sendet r' und Alice glaubt, mit Bob zu kommunizieren
    - falls $e = 1$ (mit Wahrscheinlichkeit 1/2)
      $\Rightarrow$ Mallory muss $y \equiv r' \cdot s \bmod n$ berechnen und kann das nicht

  (2) wählt y', berechnet $x' \equiv y'^2 \cdot v^{-1} \bmod n$ und sendet x' an Alice
    - falls $e = 1$ (mit Wahrscheinlichkeit 1/2)
      $\Rightarrow$ Mallory sendet y' an Alice ($y'^2 \equiv x' \cdot v \bmod n$) und Alice glaubt, mit Bob zu kommunizieren
    - falls $e = 0$ (mit Wahrscheinlichkeit 1/2)
      $\Rightarrow$ Mallory muss r' senden und kann das nicht

Identifikation (Forts.)

Zero-Knowledge-Beweis nach Fiat-Shamir (Forts.)

- Sicherheit
  - beim ersten Versuch hat Mallory eine 50:50 Chance
  - bei k=20 Versuchen hat Mallory eine Chance von ca. 1: 1 000 000 ($P = 1/2^k$)
  - n sollte 2048 Binärstellen besitzen und s sowie r nicht zu klein gewählt werden
- Zero-Knowledge
  - Bob antwortet entweder mit r oder mit y, aber nicht mit beiden Werten
  - aus den Antworten von Bob, kann nicht auf s geschlossen werden und auch nicht auf die Primfaktoren von n
  - jedes r darf nur genau einmal verwendet werden, da sich s berechnen lässt, wenn y **und** r übertragen werden: $s \equiv y \cdot r^{-1} \bmod n$
- Beispiel: gegeben für Bob $n = 77, s = 13 \Rightarrow v = 13^2 \bmod 77 = 15$
  - Bob wählt $r = 19 \Rightarrow$ sendet $x = 19^2 \bmod 77 = 53$ an Alice
  - Alice sendet $e = 1$ an Bob
  - Bob sendet $y = 19 \cdot 13 \bmod 77 = 16$ an Alice
  - Alice verifiziert, dass $y^2 \equiv x \cdot v \bmod 77$
    also $16 \cdot 16 \equiv 53 \cdot 15 \bmod 77$
    $25 = 25$

Lessons Learnt

Sicher ist sicher – einige Tipps

(1) keine eigenen Verschlüsselungsalgorithmen erfinden;
bewährte (AES, RSA, DHKE, ECDH, ECDSA) einsetzen

(2) das Prinzip von Kerkhoffs beachten

(3) immer aktuell sichere Schlüssellängen einsetzen ($\Rightarrow$ BSI)

(4) Schlüssel möglichst zufällig wählen, nicht aus Passwörtern konstruieren

(5) starke Zufallszahlen mit ausreichend hoher Entropie verwenden

(6) Schlüssel nicht zu häufig verwenden sondern regelmäßig ändern

(7) ECB-Modus nicht verwenden

(8) Initialisierungsvektoren (Nonces) nicht wiederverwenden

(9) SHA-2 oder SHA-3 als kryptografische Hashfunktion verwenden

(10) Signaturen immer verifizieren