Grundlagen der Informatik

Prof. Dr. J. Schmidt
Fakultät für Informatik

GDI – WS 2018/19

Kryptographie
Moderne Verfahren

Hochschule Rosenheim
University of Applied Sciences

Überblick

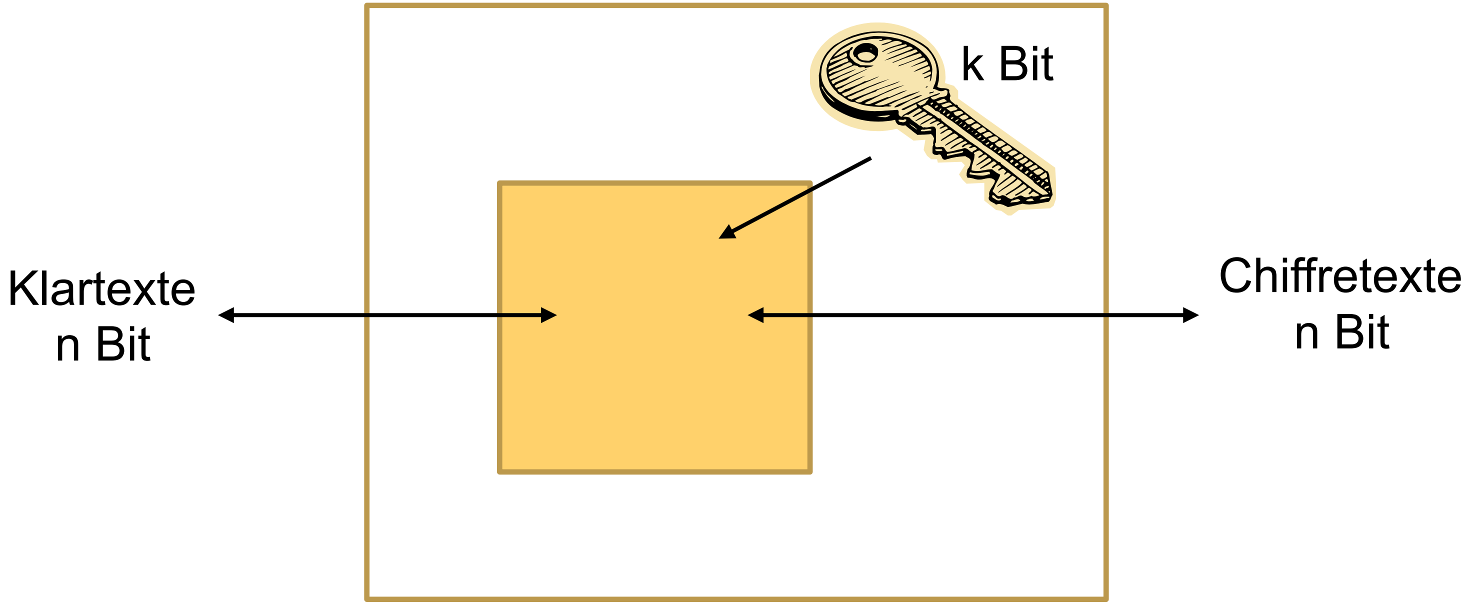
- Moderne Blockchiffren
- Asymmetrische Verschlüsselungsverfahren
  - Diffie-Hellman-Schlüsseltausch
  - RSA-Algorithmus
  - Elliptische Kurven
- Details: siehe Literatur, z.B.

  D. Wätjen. ***Kryptographie: Grundlagen, Algorithmen, Protokolle***, Spektrum Akademischer Verlag, 2. Aufl. 2008

  C. Paar, J. Pelzl und B. Preneel. ***Understanding Cryptography: A Textbook for Students and Practitioners***, Springer, 2010

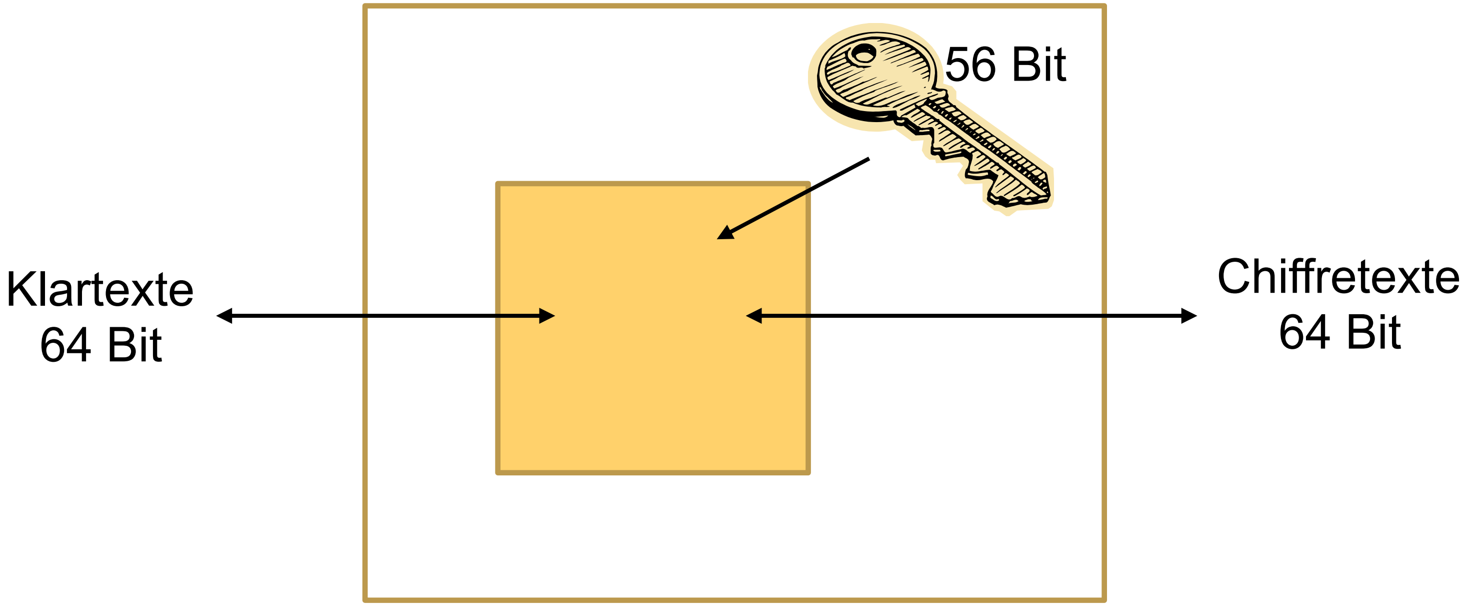
Moderne Blockchiffren

- sind symmetrische Verfahren
- die Klartexte blockweise verschlüsseln



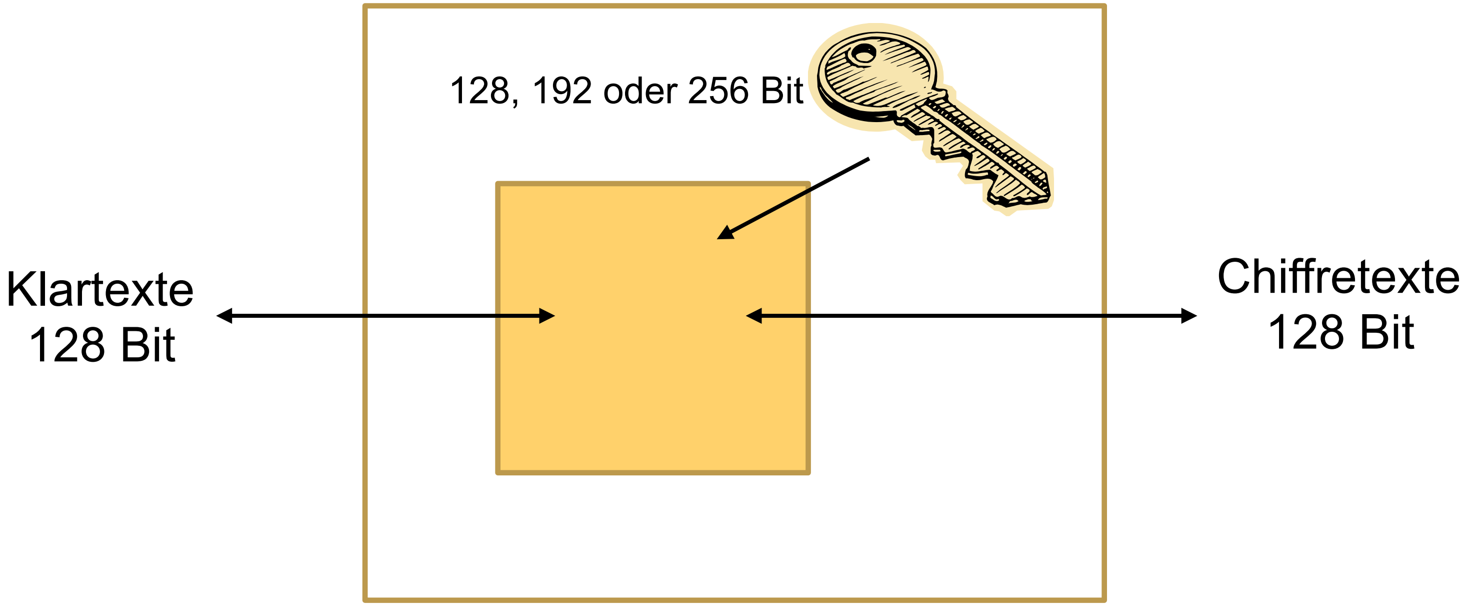
DES

- Data Encryption Standard (DES)
- 1973-77: Entwicklung und Veröffentlichung
- seither extrem weit verbreitet
- nicht mehr sicher
  - 1994 erstmals gebrochen (50 Tage auf 12 Rechnern)
  - 1998 mit Spezialchip der Electronic Frontier Foundation (EFF) weniger als 3 Tage Rechenzeit
  - 1999 DES-Challenge: 22:15h auf 100.000 PCs verteilt und EFF-Rechner
- die Variante 3DES („Triple DES“) gilt noch als sicher



AES

- Advanced Encryption Standard (AES)
- 1997: Ausschreibung eines Entwicklungswettbewerbs
- 2000/2001 AES wird Standard
  - mit Algorithmus Rijndael
  - nach den belgischen Entwicklern J. Daemen und V. Rijmen
- sicherer und effizienter als 3DES
  - ca. 3x schneller als DES
  - ca. 9x schneller als 3DES

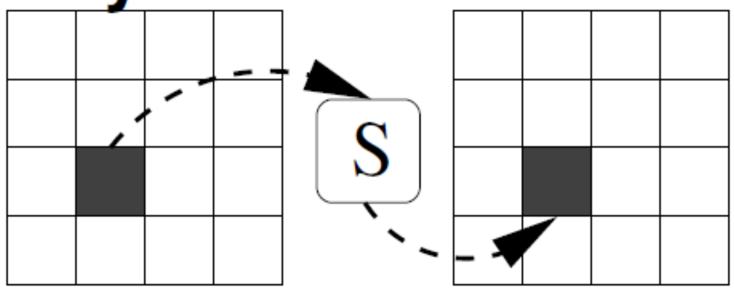


AES – Struktur

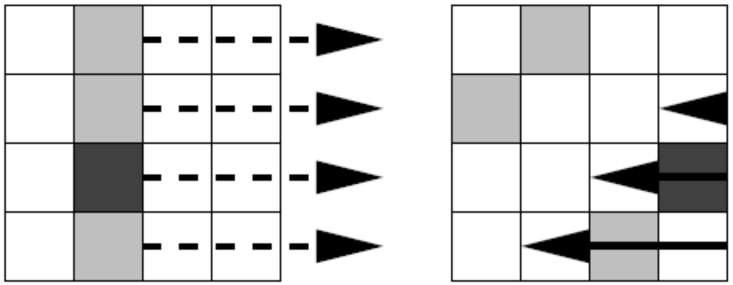
- Verschlüsselung in mehreren Runden
- Vier Basisoperationen
  - diese werden in jeder Runde kombiniert
- für jede Runde ein separater Rundenschlüssel
  - „Key Schedule“
  - generiert aus dem Chiffrier-Schlüssel (128-256 Bit)
    11-15 Rundenschlüssel (je 128 Bit)
- Zahl der Runden r abhängig von Blockgröße n und Schlüssellänge k:

r	n = 128	n = 192	n = 256
k = 128	10	12	14
k = 192	12	12	14
k = 256	14	14	14

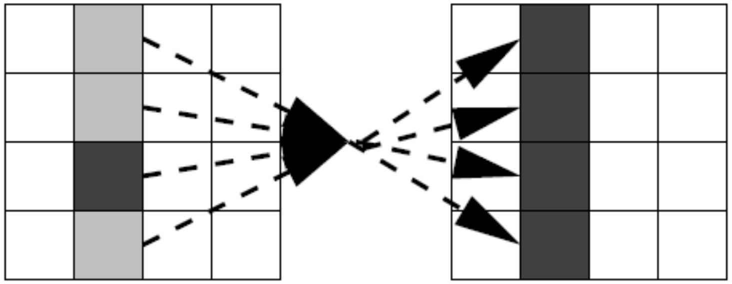
Sub Bytes



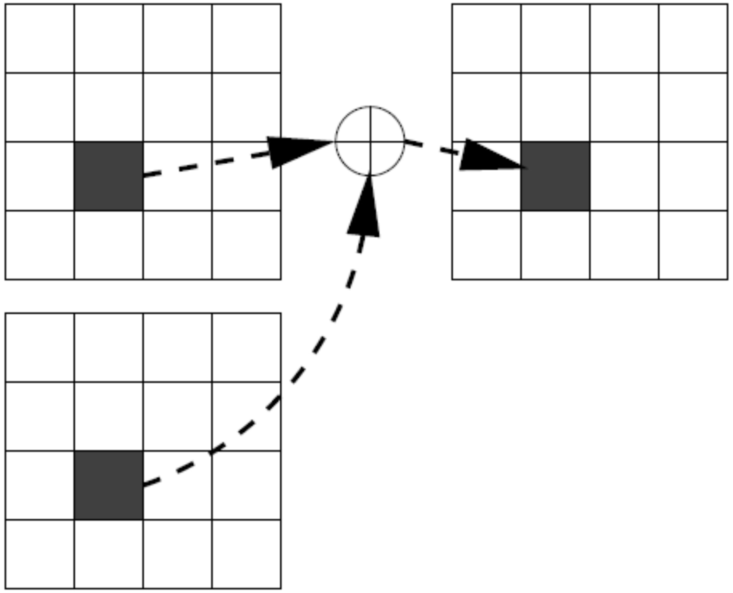
Shift Rows



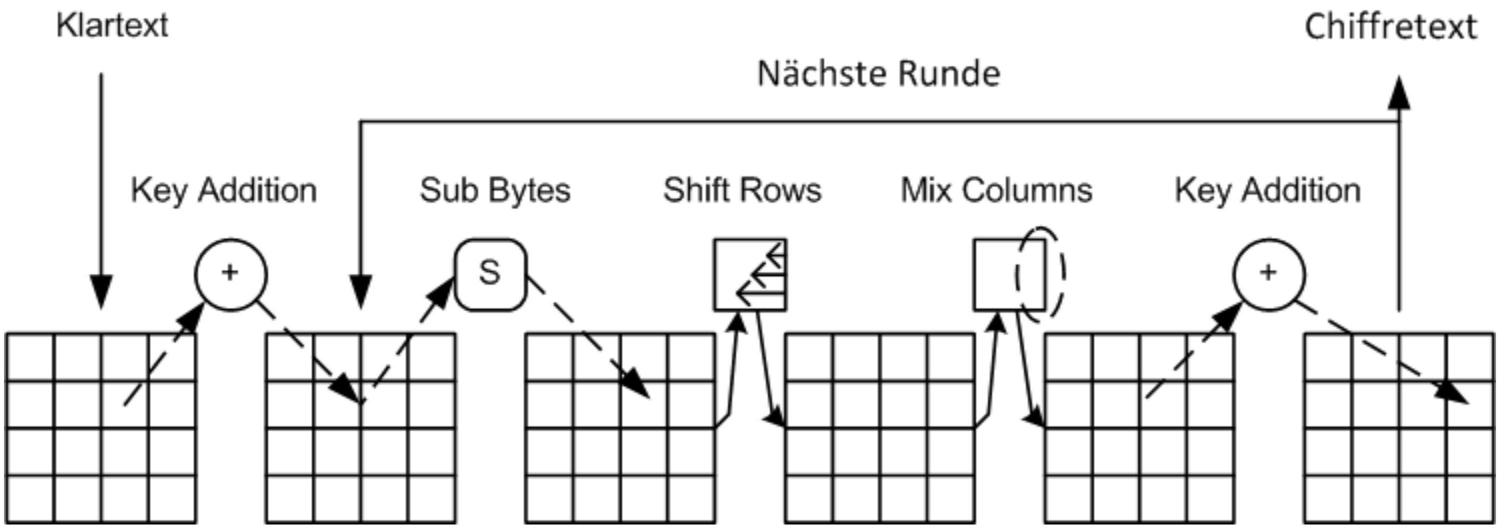
Mix Columns



Key Addition



AES – Rundenstruktur



AES – Anwendung

- verwendet z.B. in folgenden Protokollen
  - SSH (Secure Shell)
  - TLS (Transport Layer Security)
  - IPSec (Internet Protocoll Security)
- WPA2 (Wi-Fi Protected Access 2) Verschlüsselung im WLAN

Rückblick – Symmetrische Kryptosysteme (1)

- Eigenschaften symmetrischer Verschlüsselungsverfahren
  - Wer verschlüsseln kann, kann auch entschlüsseln
  - Je zwei Partner müssen einen gemeinsamen geheimen Schlüssel austauschen

Rückblick – Symmetrische Kryptosysteme (2)

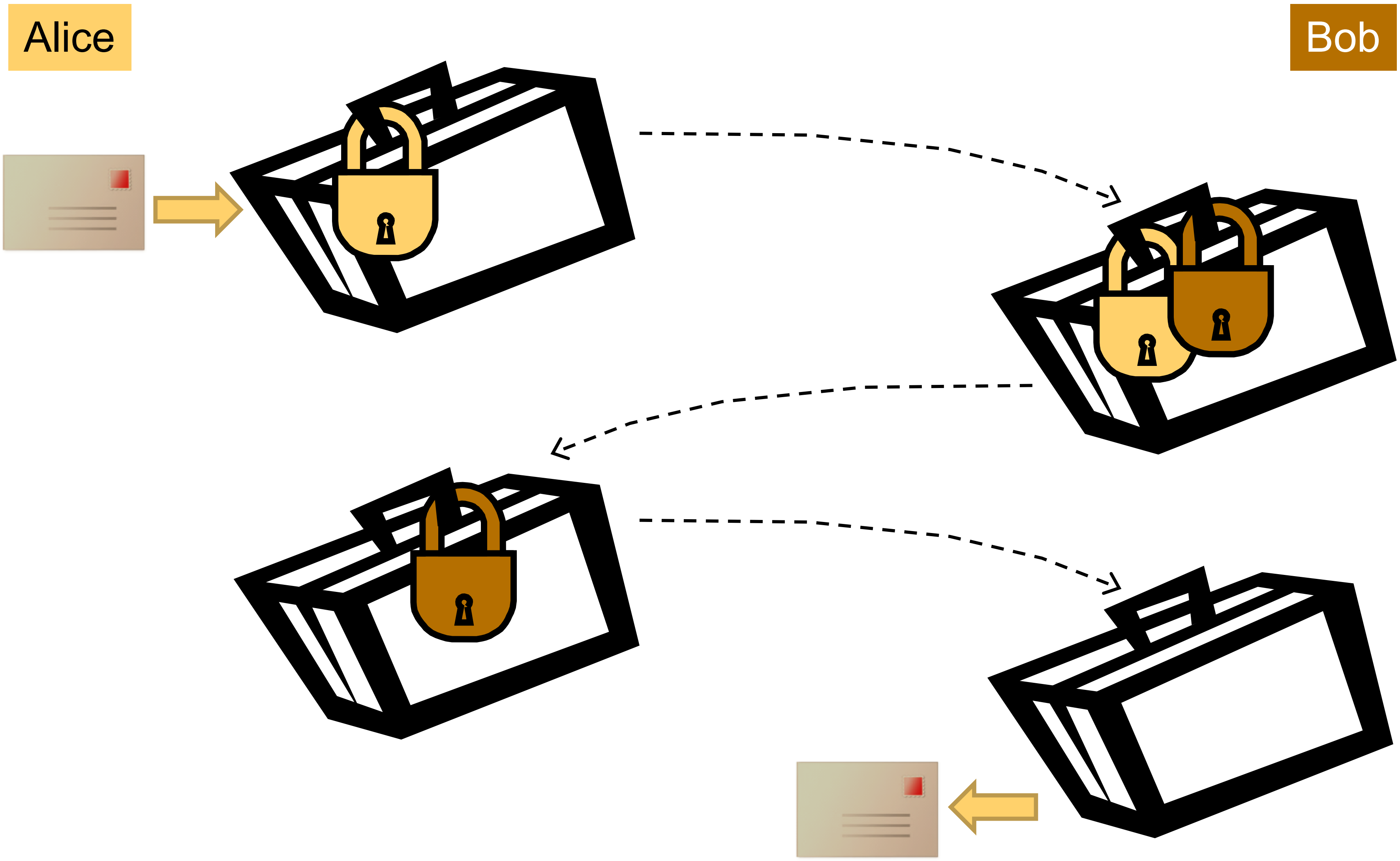
- Bewertung
  - Austausch geheimer Schlüssel
    - Sicherer Kanal notwendig
    - Oft aber offener Kanal (z.B. Bote oder Funkverbindung)
  - Schlüsselmanagement
    - Vielzahl von Schlüsseln erforderlich
    - Schwierigkeit
      - Wenn Sender und Empfänger noch nicht miteinander zu tun hatten
      - Wenn an mehrere Empfänger gleichzeitig eine Nachricht versendet werden soll
  - Authentizität ist nicht gewährleistet (identische Schlüssel der Kommunikationspartner)
- Lösung: Asymmetrische Kryptosysteme

Diffie-Hellman-Schlüsseltausch

- erstes System mit öffentlichem Schlüssel
- Diffie und Hellman 1976
- bereits 1975 von Ellis, Cocks, Williamson am britischen GCHQ entdeckt, aber geheim gehalten
- löst das Problem des Schlüsselaustauschs über einen unsicheren Kanal

- verwendet z.B. in folgenden Protokollen
  - SSH (Secure Shell)
  - TLS (Transport Layer Security)
  - IPSec (Internet Protocol Security)

Prinzip – Schlüsselaustausch



Prinzip – Schlüsselaustausch

Vorgehensweise

1. Alice verschließt Koffer mit Botschaft mit einem Vorhängeschloss, zu dem nur sie einen Schlüssel hat
2. Senden des verschlossenen Koffers an Bob
3. Bob bringt zweites Vorhängeschloss an und sendet Koffer wieder zurück
4. Alice entfernt eigenes Vorhängeschloss und sendet Koffer zurück
5. Bob entfernt eigenes Vorhängeschloss und entnimmt die Botschaft

Diffie-Hellman-Schlüsseltausch

Wähle zwei öffentliche Zahlen

- eine Primzahl p
- und eine ganze Zahl $g \in \{2, 3, \ldots, p-2\}$

1. Alice wählt zufällig eine Zahl $x_A \in \{2, 3, \ldots, p-2\}$

$$y_A = g^{x_A} \bmod p$$

   x_A bleibt geheim, y_A wird an Bob gesendet

2. Bob wählt zufällig eine Zahl $x_B \in \{2, 3, \ldots, p-2\}$

$$y_B = g^{x_B} \bmod p$$

   x_B bleibt geheim, y_B wird an Alice gesendet

3. Alice rechnet

$$k_{AB} = {y_B}^{x_A} \bmod p = (g^{x_B} \bmod p)^{x_A} \bmod p = g^{x_B x_A} \bmod p$$

4. Bob rechnet

$$k_{AB} = {y_A}^{x_B} \bmod p = (g^{x_A} \bmod p)^{x_B} \bmod p = g^{x_A x_B} \bmod p$$

Der zum Nachrichtenaustausch verwendete Schlüssel ist k_{AB}

Diffie-Hellman – Sicherheit

g sollte eine primitive Wurzel modulo p sein

- muss also die Ordnung $p-1$ haben, d.h.
- $g^{p-1} = 1 \bmod p$ und $g^a \neq 1 \bmod p$ für alle $a < p-1$

- d.h., g ist ein erzeugendes Element, es entstehen alle Elemente des Körpers außer Null
- die Anzahl solcher Elemente ist $\phi(p-1)$

- g ist genau dann eine primitive Wurzel, wenn gilt
$$g^{\frac{p-1}{r}} \neq 1 \bmod p$$
für jeden Primfaktor r von $p-1$

Erinnerung – Eulersche ϕ-Funktion

- Gibt die Anzahl der natürlichen Zahlen an
  - die kleiner als n sind
  - und keinen gemeinsamen Teiler mit n haben
  - $\phi(n) = |\{1 \leq x \leq n \mid ggT(x, n) = 1\}|$
- Berechnung (p, q sind Primzahlen $p \neq q$)
  - $\phi(p) = p - 1$ alle Zahlen von 1 bis $p - 1$ sind zu p teilerfremd
  - $\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$
  - $\phi(p^i) = p^{i-1}(p-1)$
  - $\phi(p^i q^j) = \phi(p^i)\phi(q^j) = p^{i-1}(p-1)\, q^{j-1}(q-1)$
- Beispiele
  - $\phi(5) = 4$
    - es gibt vier zu 5 teilerfremde Zahlen < 5, nämlich 1, 2, 3, 4
  - $\phi(15) = \phi(3 \cdot 5) = \phi(3)\phi(5) = 2 \cdot 4 = 8$
  - $\phi(27) = \phi(3^3) = 3^2 \cdot (3-1) = 9 \cdot 2 = 18$
    - die zu 27 teilerfremden Zahlen sind: 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26
  - $\phi(72) = \phi(2^3 \cdot 3^2) = 2^2 \cdot (2-1) \cdot 3^1 \cdot (3-1) = 4 \cdot 3 \cdot 1 \cdot 2 = 24$

Diffie-Hellman – Sicherheit

- g sollte eine primitive Wurzel modulo p sein
  - muss also die Ordnung $p - 1$ haben, d.h.
  - $g^{p-1} = 1 \bmod p$ und $g^a \neq 1 \bmod p$ für alle $a < p - 1$
  - d.h., g ist ein erzeugendes Element, es entstehen alle Elemente des Körpers außer Null
  - die Anzahl solcher Elemente ist $\phi(p - 1)$
- p sollte eine sichere Primzahl sein
  - $p = 2q + 1$, wobei q ebenfalls prim
  - sonst gibt es Nachrichten, die durch das Verfahren überhaupt nicht verändert werden: $y_A = g^{x_A} \bmod p = g$
- es gibt dann $\phi(p - 1)$ = $\phi(2q)$ = $\phi(2)\phi(q)$ = $q - 1 = \frac{p-3}{2}$ Wurzeln
  - der Körper hat p Elemente $\rightarrow$ Wahrscheinlichkeit, dass eine zufällige Zahl eine primitive Wurzel ist ca. 50%
- als sicher gelten heute Zahlen mit einer Länge ab 2000 Bit
  - p muss also größer als 2^{2000} sein $\rightarrow$ ca. 10^{602} $\rightarrow$ Primzahl mit 602 Dezimalstellen!

Diffie-Hellman – Beispiel

Wähle zwei öffentliche Zahlen

- eine Primzahl $p = 23 = 2 \cdot 11 + 1$
  $\rightarrow$ sichere Primzahl, es gibt 10 prim. Wurzeln
- und eine ganze Zahl $g \in \{2, 3, \ldots, 21\}$: $g = 5$

- 5 ist eine primitive Wurzel, da
  - $5^{\frac{22}{2}} = 5^{11} = 22 \bmod 23$ und $5^{\frac{22}{11}} = 5^2 = 25 = 2 \bmod 23$
  - 5 fortlaufend mit sich selbst multipliziert erzeugt alle Zahlen von 1 bis 22

- 2 ist **keine** primitive Wurzel, da
  - $2^{\frac{22}{2}} = 2^{11} = 1 \bmod 23$
  - 2 fortlaufend mit sich selbst multipliziert erzeugt **nicht** alle Zahlen von 1 bis 22: {2, 4, 8, 16, 9, 18, 13, 3, 6, 12, 1}

Diffie-Hellman – Beispiel

- $p = 23,\ g = 5$

1. Alice wählt zufällig eine Zahl $x_A \in \{2, 3, \dots, 21\} \rightarrow 3$
$$y_A = 5^3 \bmod 23 = 10$$
3 bleibt geheim, 10 wird an Bob gesendet

2. Bob wählt zufällig eine Zahl $x_B \in \{2, 3, \dots, 21\} \rightarrow 5$
$$y_B = 5^5 \bmod 23 = 20$$
5 bleibt geheim, 20 wird an Alice gesendet

3. Alice rechnet
$$k_{AB} = 20^3 \bmod 23 = 19$$

4. Bob rechnet
$$k_{AB} = 10^5 \bmod 23 = 19$$

Der zum Nachrichtenaustausch verwendete Schlüssel ist 19

Diffie-Hellman – Sicherheit

- Sicherheit basiert auf Verwendung einer Einwegfunktion
  - hier: diskrete Exponentiation ist einfach

$$y_A = g^{x_A} \bmod p$$

  - Umkehrung erfordert Berechnung des **diskreten Logarithmus** → sehr schwierig (zumindest glaubt man das …)
  - ein Weg, das Verfahren ohne diskreten Logarithmus zu brechen ist bisher nicht bekannt
- Einwegfunktionen
  - Injektive Funktion f: X→Y
  - y = f (x) ist für alle x∈X effizient berechenbar
  - x kann aus der Kenntnis von y nicht effizient berechnet werden
  - D.h. Umkehrfunktion x = f^{-1}(y) kann nur mit unrealistischem Aufwand ermittelt werden

Einwegfunktionen

- ob Einwegfunktionen überhaupt existieren ist unbekannt!
  - ein Beweis dafür würde den Beweis von $P \neq NP$ einschließen (→ Vorlesung Theoretische Informatik; umgekehrt gilt das nicht)

- Beispiele für Funktionen, die die Bedingung evtl. erfüllen
  - diskrete Exponentiation
  - (kryptographische) Hash-Funktionen
    - MD5 (Message Digest, 128 Bit Länge)
    - SHA-1 (Secure Hash Algorithm, 160 Bit Länge)
    - SHA-2/SHA-3 (224 bis 512 Bit)
    - typische Anwendung: Verschlüsselung von Passwörtern
    - MD5 und SHA-1 gelten nicht mehr als sicher
  - Primzahlen
    - Multiplikation ist einfach
    - Faktorisierung ist schwierig

Falltürfunktionen

- Spezialfall von Einwegfunktionen
- Unter Verwendung einer Zusatzinformation
  - (eines Schlüssels)
  - sind die Umkehrfunktionen effizient berechenbar

- Beispiel: Faktorisierung
  - einfach, wenn einer der beiden Faktoren bekannt
  - → RSA

RSA-Algorithmus

- 1978 von R. **R**ivest, A. **S**hamir und L. **A**dleman entwickelt
- Basiert auf der Annahme, dass
  - die Faktorisierung großer Zahlen (Zerlegung in Primfaktoren) sehr aufwändig ist
  - das Erzeugen einer solch großen Zahl durch die Multiplikation zweier Primzahlen sehr einfach ist

RSA-Algorithmus – Schlüsselgenerierung

1. Auswahl zweier großer Primzahlen p und q

2. Berechnung des RSA-Moduls n

$$n = pq$$

- n sollte mindestens 500 (dezimale) Stellen haben
- Faktorisierung auch mit Super-Computer nicht effizient möglich
- Bei Kenntnis von p und q ist die Faktorisierung durch eine Division möglich

3. Berechnung der Eulerschen Funktion von n

$$\phi(n) = (p - 1)\,(q - 1)$$

4. Auswahl eines Verschlüsselungsexponenten *c*

- *c* ist kleiner als $\phi(n)$

$$1 < c < \phi(n)$$

- *c* hat keinen gemeinsamen Teiler mit der Eulerschen Funktion

$$\mathrm{ggT}(c, \phi(n)) = 1$$

- Zahlenpaar (*c*, *n*) bilden den **öffentlichen** Schlüssel

5. Berechnung des Entschlüsselungsexponenten *d* als modulare Inverse von *c* bzgl. $\phi(n)$

$$c \cdot d \mod \phi(n) = 1$$

- Mit erweitertem euklidischen Algorithmus oder Satz von Euler
- d ist der **private** Schlüssel

RSA-Algorithmus – Senden und Empfangen

- „Alice möchte Nachricht an Bob senden“
  - Nachschlagen des öffentlichen Schlüssels von Bob in Schlüsselverzeichnis

    $$(n_{\mathrm{Bob}}, c_{\mathrm{Bob}})$$

  - Aufteilung der Nachricht in Abschnitte x_1, x_2, x_3, …
  - Berechnung der verschlüsselten Abschnitte

    $$y_{\mathrm{i}} = x_{\mathrm{i}}^{\,c_{\mathrm{Bob}}} \bmod n_{\mathrm{Bob}}$$

  - Übermittlung der y_i
  - Entschlüsselung durch Bob unter Verwendung des nur ihm bekannten privaten Schlüssel d_{Bob}

    $$x_{\mathrm{i}} = y_{\mathrm{i}}^{\,d_{\mathrm{Bob}}} \bmod n_{\mathrm{Bob}}$$

RSA – Beweis

- Satz von Euler

$$a^{\phi(n)} \bmod n = 1 \qquad \text{wenn ggT}(a, n) = 1$$

- RSA

$$x^{cd} \bmod n$$

$$cd \bmod \phi(n) = 1 \qquad \Rightarrow cd = 1 + k\phi(n)$$

$$x^{cd} \bmod n =$$
$$x^{1 + k\phi(n)} \bmod n =$$
$$x\, x^{k\phi(n)} \bmod n =$$
$$x\, (x^{\phi(n)})^{k} \bmod n = x$$

Beispiel RSA-Algorithmus (1)

- Alice möchte an Bob eine verschlüsselte Nachricht senden
  - Nur die 26 lateinischen Buchstaben werden verwendet
  - Numerische Darstellung der Buchstaben
    - Jedem Buchstaben wird seine Position im Alphabet zugeordnet (A → 1, …, Z → 26)
  - Aufteilung der Nachricht in Blöcke
    - Ein Block enthält ein Zeichen

Beispiel RSA-Algorithmus (2)

1. Auswahl zweier Primzahlen
   - p = 5 und q = 11
2. Berechnung des RSA-Moduls n
   - $n = 5 \cdot 11 = 55$
3. Berechnung der Eulerschen Funktion von n
   - $\phi(n) = (p - 1)(q - 1) = 4 \cdot 10 = 40$
4. Auswahl eines Verschlüsselungsexponenten c
   - z.B. c = 3, da größter gemeinsamer Teiler von c und $\phi(n) = 1$
5. Berechnung des Entschlüsselungsexponenten d
   - mit $d = c^{-1} = c^{\phi(\phi(n))-1} \bmod \phi(n)$
   - $d = 3^{\phi(40)-1} \bmod 40 = 3^{15} \bmod 40 = 27$
   - $\phi(40) = \phi(2^3 \cdot 5) = 2^2 \cdot 1 \cdot 4 = 16$

Beispiel RSA-Algorithmus (3)

- Verschlüsselung des Textes `CLEO`
  - Bildung der numerischen Darstellung
    ➔ 3, 12, 5, 15
  - Verschlüsselung mit öffentlichem Schlüssel c = 3

    `C`: $y_1 = 3^3 \bmod 55 = 27$
    `L`: $y_2 = 12^3 \bmod 55 = 1728 \bmod 55 = 23$
    `E`: $y_3 = 5^3 \bmod 55 = 125 \bmod 55 = 15$
    `O`: $y_4 = 15^3 \bmod 55 = 3375 \bmod 55 = 20$
  - Die Zahlenfolge 27, 23, 15, 20 wird gesendet

- Entschlüsselung (der Zahlenfolge 27, 23, 15, 20)
  - Empfänger verwendet seinen geheimen Schlüssel d = 27

$x_1 = 27^{27} \bmod 55 = 3 \rightarrow$ C
$x_2 = 23^{27} \bmod 55 = 12 \rightarrow$ L
$x_3 = 15^{27} \bmod 55 = 5 \rightarrow$ E
$x_4 = 20^{27} \bmod 55 = 15 \rightarrow$ O

RSA – Anmerkungen

- ca. 1000x langsamer als gängige symmetrische Verschlüsselungsverfahren (z.B. AES, 3DES)

- daher: Anwendung als hybrides Verfahren
  - RSA zur Verschlüsselung eines gemeinsamen (symmetrischen) Schlüssels
  - Übertragung des verschlüsselten symmetrischen Schlüssels
  - eigentlicher Datenaustausch mit symmetrischem Verfahren

- Anwendungsbeispiele
  - Protokolle SSH, TLS (in https)
  - RFID-Chip in deutschem Reisepass

RSA Factoring Challenge

- Wettbewerb der Firma RSA Security
  - sollte Sicherheit der RSA-Verschlüsselung zeigen
  - gestartet 18.3.1991
  - eingestellt 2007
- gegeben: Zahl entstanden als Produkt aus genau zwei Primzahlen
- gesucht: die beiden Primfaktoren

RSA Factoring Challenge (Auszug)

RSA Zahl	#Stellen dezimal	#Stellen binär	Preisgeld	Datum Faktorisierung	Anmerkungen
RSA-100	100	330	**$1.000**	1.4.1991	Lenstra, Uni Amsterdam, wenige Tage
RSA-110	110	364	**$4.429**	14.4.1992	Lenstra, Uni Amsterdam, 1 Monat
RSA-155	155	512	**$9.383**	22.8.1999	te Riele et al., CWI Amsterdam, 8000 MIPS-Jahre
RSA-576	174	576	**$10.000**	3.12.2003	Franke et al., Uni Bonn
RSA-210	210	696	-	26.9.2013	Ryan Propper
RSA-220	220	729	-	13.5.2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé, P. Zimmermann, Australian National University, ca. 370 CPU-Jahre (Xeon E5-2650, 2GHz)
RSA-230	230	762	-	15.8.2018	Samuel S. Gross, Noblis Inc.
RSA-640	193	640	**$20.000**	2.11.2005	Franke et al., Uni Bonn, 5 Monate auf 80 AMD Opteron 2,2 GHz
RSA-704	212	704	$30.000	2.7.2012	S. Bai, E. Thomé, P. Zimmermann, Australian National University, ca. 14 Monate
RSA-768	232	768	$50.000	12.12.2009	Kleinjung (Lausanne) et al. 2000 CPU-Jahre (single-core AMD Opteron 2,2 GHz) http://eprint.iacr.org/2010/006.pdf
RSA-1024	309	1024	$100.000	-	ca. 1000x schwerer als RSA-768
RSA-1536	463	1536	$150.000	-	
RSA-2048	617	2048	$200.000	-	

Aufgabe

- Verschlüsseln Sie folgende Nachricht mit dem RSA-Algorithmus:

```
DAS_IST_GEHEIM
```

- Verwenden Sie dabei
  - Kodierung
    - _ = 00, A = 01, B = 02, ..., Z = 26
  - p = 47, q = 79 und c = 37
  - Zerlegung der Nachricht in vier Ziffern (= 2 Zeichen des Texts) lange Teilstücke

Angriffe auf Kryptosysteme (1)

- Öffentliche Schlüssel
- Verwendung von zentralem System zur Schlüsselverwaltung „Key Server“
- Anfällig gegen sog. „Man-in-the-Middle-Angriff“

Angriffe auf Kryptosysteme (2)

- Man-in-the-Middle-Angriff
  - Angreifer nistet sich im Key Server ein
  - Gibt bei der Anfrage nach einem öffentlichen Schlüssel seinen eigenen Schlüssel zurück
  - Gesendete Nachricht wird abgefangen
  - Wird mit dem eigenen Schlüssel entschlüsselt

Angriffe auf Kryptosysteme (3)

- Man-in-the-Middle-Angriff (Fortsetzung)
  - Nachricht wird mit öffentlichem Schlüssel des eigentlichen Empfängers neu verschlüsselt
    - Änderung der ursprünglichen Nachricht leicht möglich
  - Verschlüsselte Nachricht wird weitergesendet
  - Empfänger bemerkt den Angriff nicht
    - Annahme, dass Nachricht vom ursprünglichen Sender kommt
- Ansätze gegen Man-in-the-Middle-Angriff
  - Digitale Unterschriften
  - und das „Web of Trust“

Web of Trust (1)

- Idee
  - Echtheit von digitalen Schlüsseln wird durch ein Netz von gegenseitigen Bestätigungen gesichert
- Zertifikat
  - Digitale Signatur auf einen Schlüssel
  - Abgabe durch eine Person, die auch am Web of Trust teilnimmt
  - Wenn diese Person sich über die Identität des Schlüsselinhabers versichert hat

Web of Trust (2)

- Zertifizierungsstellen
  - Certification Authorities (CA)
  - Schlüssel können auch durch Signaturen entsprechender Zertifizierungsstellen beglaubigt werden
- Vertrauenswürdige Schlüsselquellen
  - Einrichtung dezentraler Sammelstellen für öffentliche Schlüssel (Key Server)
  - Sammlung von Schlüsseln vertrauenswürdiger Schlüsselquellen

Beispiel Web of Trust (1)

- Alice
  - erzeugt ein Schlüsselpaar (öffentlicher und privater Schlüssel)
  - signiert das Schlüsselpaar
  - schickt öffentlichen Schlüssel an Schlüsselserver
- Bob
  - möchte mit Alice verschlüsselt kommunizieren
  - besorgt sich Alice Schlüssel von Schlüsselserver
  - fragt Alice nach Details ihres öffentlichen Schlüssels
    - ID, Länge, Typ oder digitaler Fingerabdruck
    - Persönliche Kontakt (Treffen, Telefon, …)

Beispiel Web of Trust (2)

- Bob (Fortsetzung)
  - vergleicht die Daten mit den denen des vom Schlüsselserver erhaltenen Schlüssel
  - signiert den öffentlichen Schlüssel von Alice mit seinem privaten Schlüssel
    - falls Daten übereinstimmen
  - schickt diese Signatur wieder an den Schlüsselserver
- Karl
  - möchte mit Alice verschlüsselt kommunizieren
  - besorgt sich den öffentlichen Schlüssel von Alice
  - stellt fest, dass Bob den Schlüssel bereits überprüft hat
  - wenn Karl Bob vertraut, vertraut er dem Schlüssel von Alice
    - muss keine Prüfung von Alice durchführen

Digitale Unterschrift (1)

- Sicherung der Authentizität

Verfahren:

- Berechnung eines Zwischenergebnisses s
  - aus der zu übermittelnden Botschaft x
  - unter Verwendung des eigenen privaten Schlüssels d_{Alice}

  $s = x^{d_{Alice}} \bmod n$

- Verschlüsselung des Zwischenergebnisses s
  - mit dem öffentlichen Schlüssels des Kommunikationspartners c_{Bob}

  $y = s^{c_{Bob}} \bmod n$

Digitale Unterschrift (2)

- Sicherung der Authentizität

Verfahren (Fortsetzung):

- Nach Empfang der signierten Nachricht y
  - Anwendung des privaten Schlüssels
  - Resultat ist das Zwischenergebnis s

    $s = y^{d_{Bob}} \bmod n$

- Empfänger schlägt öffentlichen Schlüssel des Kommunikationspartners im Schlüsselverzeichnis nach
  - Anwendung auf das Zwischenergebnis s

    $x = s^{c_{Alice}} \bmod n$

- „Vernünftiges“ Ergebnis
  - = sicher, dass die Nachricht vom richtigen Absender kommt

- in der Praxis:
  - erzeugen eines Hash-Werts (z.B. mit SHA-3)
  - signieren dieses Werts

Elliptische Kurven – Elliptic Curve Cryptography (ECC)

- Unabhängig voneinander entdeckt von N. Koblitz (1987) und V. Miller (1987)
- Public-Key Verfahren
- Mittlerweile als Standard etabliert (z.B. IPsec, TLS)

- Vorteil gegenüber RSA:
  - Als Angriffsmöglichkeit bleibt im Prinzip nur die Berechnung des diskreten Logarithmus
    - diese ist bei ECC weniger effizient als bei RSA
  - Daher höhere Sicherheit bereits bei kleinen Schlüssellängen
    1024 Bit RSA ≈ 160 Bit ECC
    3072 Bit RSA ≈ 256 Bit ECC

Elliptische Kurve – Definition

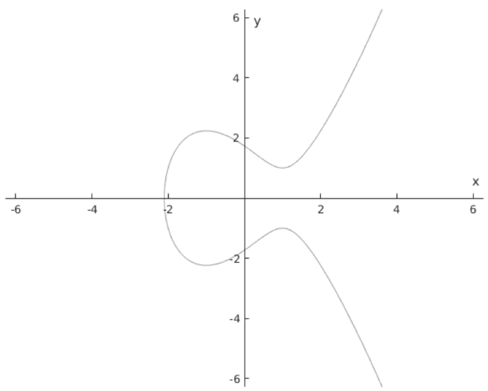
Elliptische Kurve ≠ Ellipse!

Elliptische Kurve: Alle Punkte (x, y), die folgende Gleichung erfüllen:

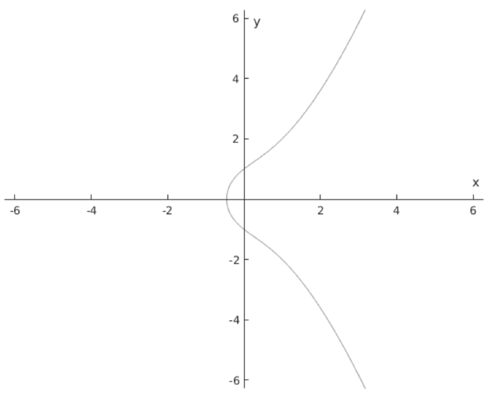
$$y^2 = x^3 + ax + b$$

mit a, b, x, y aus einem beliebigem Körper (mit mindestens 4 Elementen) und $4a^3 + 27b^2 \neq 0$

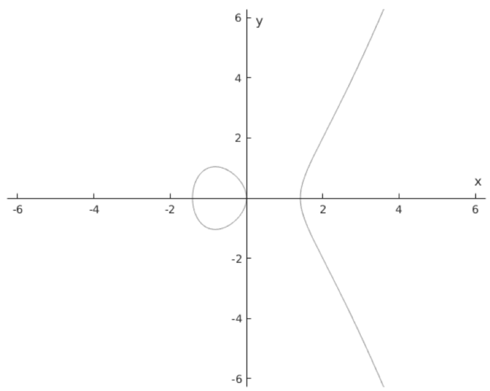
Beispiele (Plots über dem Körper der reellen Zahlen):



$y^2 = x^3 - 3x + 3$



$y^2 = x^3 + 2x + 1$



$y^2 = x^3 - 2x$

Kryptographie: verwende endlichen Körper $\mathbb{F}_q$ mit q = p^i Elementen, p prim, i ∈ $\{1, 2, 3, \ldots\}$ (i $= 1 \rightarrow$ rechne modulo p)

ECC – Womit wird gerechnet?

- Statt „normaler“ Zahlen: verwende Punkte $P = (x, y)$ aus $\mathbb{F}_q$, die die Gleichung erfülle, rechne mod Primzahl p
- → Definition einer kommutativen Gruppe (abgeschlossen, assoziativ, Neutralelement, Inverse)

- **Operation** „+“: $P_3 = P_1 + P_2 = (x_1, y_1) + (x_2, y_2)$ (dieses Symbol ist willkürlich!), für die gilt:

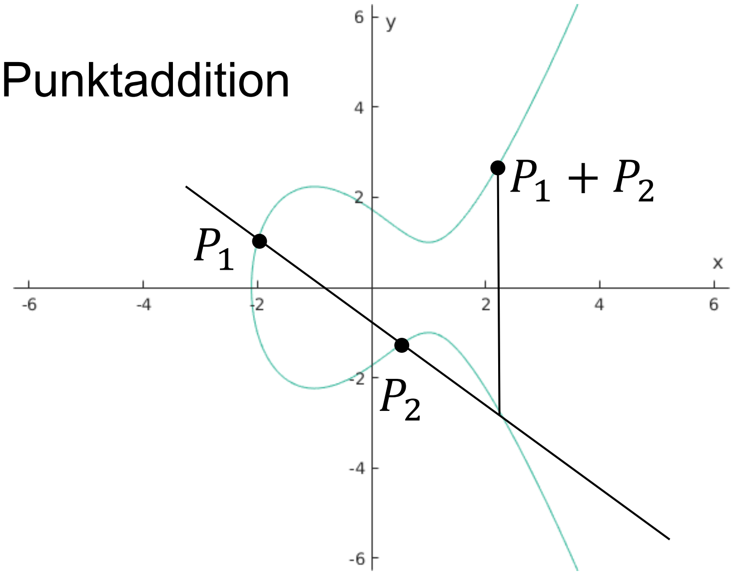
$$\begin{aligned} x_3 &= s^2 - x_1 - x_2 && \mod p \\ y_3 &= s(x_1 - x_3) - y_1 && \mod p \end{aligned}$$

mit $s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p & \text{wenn } P_1 \neq P_2 \text{ (Punktaddition)} \\ \frac{3x_1^2 + a}{2y_1} \mod p & \text{wenn } P_1 = P_2 \text{ (Punktverdopplung)} \end{cases}$

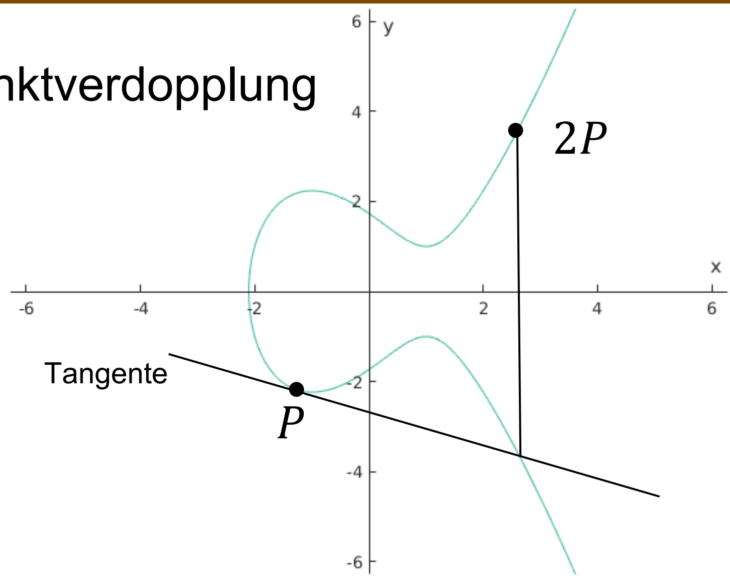
- **neutrales Element** $\mathcal{O}$ mit
$$P + \mathcal{O} = \mathcal{O} + P = P$$
(ein unendlich weit entfernter Punkt in Richtung der y-Achse)

- **Inverse** zu $P = (x, y)$ ist $-P = (x, -y)$

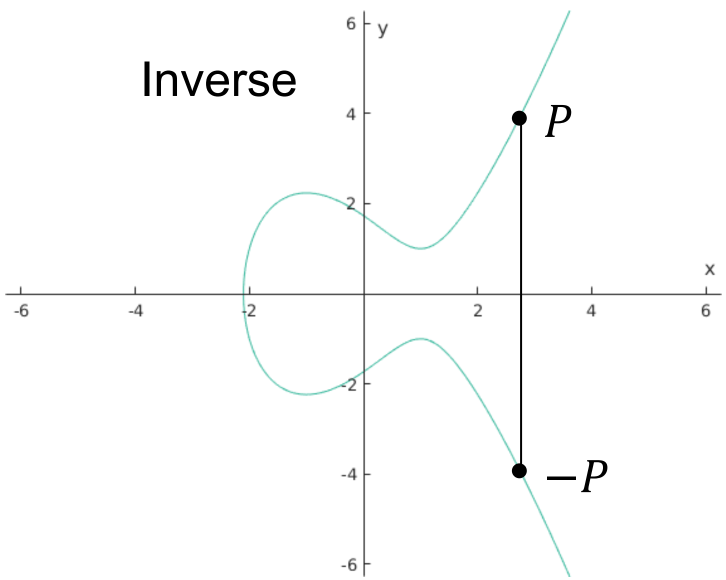
Punktaddition



Punktverdopplung



Inverse



ECC – Welche Punkte liegen auf der Kurve?

- In $\mathbb{F}_p$ (p prim): Rechnung mod p!
- Einsetzen aller Punkte x in $y^2 = x^3 + ax + b$
- Gleichung ist erfüllt genau für die *quadratischen Reste* R_p
  - Das sind Zahlen $c = x^3 + ax + b$ für die gilt $c^{\frac{p-1}{2}} \bmod p = 1$
- Für alle Elemente aus R_p: Berechnung der Quadratwurzel

- Berechnung der Wurzel ist einfach, wenn gilt $4 \mid (p + 1)$
  - Für $y^2 \bmod p = c$ lauten die Lösungen dann:
  - $y_1 = c^{\frac{p+1}{4}}$ und $y_2 = p - y_1$

- In anderen Fällen: probabilistischer Algorithmus, siehe (Wätjen 2008, Algorithmus 9.1)

- Abschätzung Anzahl Elemente N der Kurve:

$$p + 1 - 2\sqrt{p} \ \leq \ N \ \leq p + 1 + 2\sqrt{p}$$

eine Kurve besteht also aus ca. p Elementen

- Prüfe alle Zahlen $x \in \{0, 1, 2, \dots, 10\}$ ob y^2 quadratische Reste (d.h. in R_{11}) sind
- Bestimme die Quadratwurzel

x	$y^2 = x^3 + 3x + 9 \bmod 11$	y^2 in R_{11}?	y
0	9	✓	3, 8
1	2	–	
2	1	✓	1, 10
3	1	✓	1, 10
4	8	–	
5	6	–	
6	1	✓	1, 10
7	10	–	
8	6	–	
9	6	–	
10	5	✓	4, 7

Die Kurve enthält also insgesamt 11 Punkte:
Die 10 aus der Tabelle und den Punkt $\mathcal{O}$

Beispiel nach Wätjen, 2008

Wähle öffentlich

- eine Primzahl p
- eine elliptische Kurve $E\colon y^2 = x^3 + ax + b$ mit N Elementen
- ein primitives (= erzeugendes = hier beliebiges) Element $g = (x_g,\ y_g) \in E$

1. Alice wählt zufällig eine Zahl $x_A \in \{2, 3, \dots, N-1\}$, addiere g x_A mal:

$$y_A = g + g + \cdots + g = x_A g \bmod p$$

   x_A bleibt geheim, y_A wird an Bob gesendet

2. Bob wählt zufällig eine Zahl $x_B \in \{2, 3, \dots, N-1\}$

$$y_B = g + g + \cdots + g = x_B g \bmod p$$

   x_B bleibt geheim, y_B wird an Alice gesendet

3. Alice rechnet

$$k_{AB} = x_A y_B \bmod p = x_A x_B g \bmod p$$

4. Bob rechnet

$$k_{AB} = x_B y_A \bmod p = x_B x_A g \bmod p$$

Da in einer kommutativen Gruppe gerechnet wird, ist das Ergebnis identisch.
Der zum Nachrichtenaustausch verwendete Schlüssel ist k_{AB}

ECC-Diffie-Hellman – Beispiel

$p = 11, y^2 = x^3 + 3x + 9, g = (0,8)$

1. Alice wählt zufällig eine Zahl $x_A \in \{2, 3, \ldots, 10\} \rightarrow 3$

   $$y_A = (0,8) + (0,8) + (0,8) \bmod 11 = (3{,}10) + (0{,}8) = (6, 10)$$

   3 bleibt geheim, $(6, 10)$ wird an Bob gesendet

2. Bob wählt zufällig eine Zahl $x_B \in \{2, 3, \ldots, 10\} \rightarrow 2$

   $$y_B = (0,8) + (0,8) \bmod 11 = (3, 10)$$

   2 bleibt geheim, $(3, 10)$ wird an Alice gesendet

3. Alice rechnet

   $$k_{AB} = 3 \cdot (3, 10) \bmod 11 = (2, 10)$$

4. Bob rechnet

   $$k_{AB} = 2 \cdot (6, 10) \bmod 11 = (2, 10)$$

Der zum Nachrichtenaustausch verwendete Schlüssel ist $(2, 10)$

- Um das Verfahren zu brechen, muss man x_A bzw. x_B bestimmen
  - Anschaulich sind dies die Anzahl der Sprünge auf der Kurve vom Start- bis zum Endpunkt
  - Dies entspricht dem diskreten Logarithmus;
    die Formulierung mit „+“ sieht nur ungewohnt aus

- Auf diese Weise lassen sich auch andere Verschlüsselungsverfahren auf elliptische Kurven umstellen: Rechne statt mit „normalen“ Zahlen mit den Punkten der Kurve

- Die Sicherheit hängt auch von der verwendeten Kurve ab
  Beispiel: Curve 25519 (Bernstein, 2005)
  - verwendet für Diffie-Hellman
  - $p = 2^{255} - 19,\ y^2 = x^3 + 486662x^2 + x,\quad g = (9, y)$
  - (zu dieser Kurve existiert eine isomorphe Kurve als sog. *kurze Weierstraß-Gleichung*, die dann die Form $y^2 = x^3 + ax + b$ hat.)

Kurzfassung – Was sollte man derzeit verwenden?

- Hashing:
  - SHA-2 (als SHA-256, SHA-384 oder SHA-512)
  - Nachfolger SHA-3
- Symmetrische Verfahren:
  - AES-256,
  - Betriebsmodus GCA (Galois Counter Mode)
- Asymmetrische Verfahren
  - RSA mit 2048 Bit, für mittelfristige Sicherheit 3072 Bit
  - ECC mit 256 Bit (z.B. Curve 25519)

Die Zukunft

- Quantencomputer
  - alle aktuell verwendeten Public-Key Verfahren brechen zusammen
    - Shors-Algorithmus (1994): ermöglicht effiziente Primfaktorisierung und diskrete Logarithmen
    - betrifft in erster Linie Schlüsselaustausch und digitale Unterschriften
  - AES bleibt sicher

- Post-Quanten-Kryptographie erforderlich
  siehe z.B. https://pqcrypto.org/