Geheim bleibt geheim: Computeralgebra und Verschlüsselung mit RSA

Prof. Dr. Wolfram Koepf

Universität Kassel
http://www.mathematik.uni-kassel.de/~koepf

Nordhessischer Tag der Mathematik
16. Februar 2007
Universität Kassel

UNIKASSEL
VERSITÄT

Inhaltsangabe

- Was ist Computeralgebra?
- Was ist Kryptologie?
- Ein Primzahltest
- Das RSA-Verfahren

Inhaltsangabe

- Was ist Computeralgebra?
- Was ist Kryptologie?
- Ein Primzahltest
- Das RSA-Verfahren

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Was ist Computeralgebra?

General-Purpose Computeralgebrasysteme

- Computeralgebrasysteme rechnen mit Zahlen, aber auch mit Symbolen.
- Bekannte Computeralgebrasysteme sind
  - Reduce (1968)
  - Macsyma, Maxima (1971)
  - Maple (1980)
  - *Mathematica* (1988)
  - Derive (1989)
  - MuPAD (1990)
  - Axiom (1992)
- Computeralgebrasysteme gibt es auch als Taschenrechner von Casio und Texas Instruments.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Fähigkeiten von Computeralgebrasystemen

Computeralgebrasysteme können . . .

- wie ein Taschenrechner rechnen (*Mathematica*)
- mit beliebig großen ganzen Zahlen und Brüchen rechnen
- größte gemeinsame Teiler finden und Primzahlen identifizieren
- Polynome ausmultiplizieren und faktorisieren
- Gleichungen und Gleichungssysteme lösen
- graphische Darstellungen erzeugen
- Grenzwerte, Ableitungen, Integrale berechnen
- Vereinfachungen durchführen
- und stellen eine Programmiersprache zur Verfügung.

Inhaltsangabe

Was ist Kryptologie?

Alles Wissen

- Am 10. Januar 2007 war Verborgene Welten das Thema der Sendung Alles Wissen im dritten Fernsehprogramm des HR.
- Für einen Beitrag zu dieser Sendung wurde auch ich interviewt, und zwar zum Thema Kryptologie.
- Als kurzen Einblick in dieses aktuelle Forschungsgebiet sehen wir uns den fünfminütigen Beitrag über Kryptologie an.

Wie geht es im Vortrag weiter?

- In der Folge werde ich einige wesentliche Aspekte asymmetrischer Kryptosysteme vorstellen.

Was ist Kryptologie?

Alles Wissen

- Am 10. Januar 2007 war Verborgene Welten das Thema der Sendung Alles Wissen im dritten Fernsehprogramm des HR.
- Für einen Beitrag zu dieser Sendung wurde auch ich interviewt, und zwar zum Thema Kryptologie.
- Als kurzen Einblick in dieses aktuelle Forschungsgebiet sehen wir uns den fünfminütigen Beitrag über Kryptologie an.

Wie geht es im Vortrag weiter?

- In der Folge werde ich einige wesentliche Aspekte asymmetrischer Kryptosysteme vorstellen.

Was ist Kryptologie?

Alles Wissen

- Am 10. Januar 2007 war Verborgene Welten das Thema der Sendung Alles Wissen im dritten Fernsehprogramm des HR.
- Für einen Beitrag zu dieser Sendung wurde auch ich interviewt, und zwar zum Thema Kryptologie.
- Als kurzen Einblick in dieses aktuelle Forschungsgebiet sehen wir uns den fünfminütigen Beitrag über Kryptologie an.

Wie geht es im Vortrag weiter?

- In der Folge werde ich einige wesentliche Aspekte asymmetrischer Kryptosysteme vorstellen.

Was ist Kryptologie?

Alles Wissen

- Am 10. Januar 2007 war Verborgene Welten das Thema der Sendung Alles Wissen im dritten Fernsehprogramm des HR.
- Für einen Beitrag zu dieser Sendung wurde auch ich interviewt, und zwar zum Thema Kryptologie.
- Als kurzen Einblick in dieses aktuelle Forschungsgebiet sehen wir uns den fünfminütigen Beitrag über Kryptologie an.

Wie geht es im Vortrag weiter?

- In der Folge werde ich einige wesentliche Aspekte asymmetrischer Kryptosysteme vorstellen.

Inhaltsangabe

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr.
  Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären:
  $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Modulares Rechnen

Beispiel

- Beim Rechnen mit Uhrzeiten rechnen wir modulo 24.
- Beispielsweise ist 21 Uhr + 8 Stunden $\equiv$ 29 Uhr $\equiv$ 5 Uhr. Wir schreiben hierfür

$$21 + 8 \equiv 5 \pmod{24}.$$

- Ebenso kann man modulare Multiplikationen erklären: $21 \cdot 5 \equiv 105 \equiv 9$. Wir schreiben hierfür

$$21 \cdot 5 \equiv 9 \pmod{24}.$$

Modulares Inverses

- Hat p keinen gemeinsamen Teiler mit $x \in \mathbb{Z}$, so gibt es zu x ein multiplikatives Inverses $y \in \mathbb{Z}$ mit $x \cdot y \equiv 1 \pmod{p}$.

Ein Primzahltest

Kleiner Satz von Fermat

Für eine Primzahl $p \in \mathbb{P}$ und $a \in \mathbb{Z}$ gilt

$$a^p \equiv a \pmod{p} .$$

Fermattest

Ist diese Beziehung für eine Zahl $a \in \mathbb{Z}$ nicht erfüllt, so kann also p keine Primzahl sein!

Ein Primzahltest

Kleiner Satz von Fermat

Für eine Primzahl $p \in \mathbb{P}$ und $a \in \mathbb{Z}$ gilt

$$a^p \equiv a \pmod{p} .$$

Fermattest

Ist diese Beziehung für eine Zahl $a \in \mathbb{Z}$ nicht erfüllt, so kann also p keine Primzahl sein!

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Effiziente Berechnung von Potenzen

Divide- und Conquer-Algorithmus

- Um den Fermattest durchführen zu können, müssen also sehr effizient modulare Potenzen berechnet werden.
- Die modulare Potenz $a^n \pmod p$ berechnet man am besten durch Zurückführen auf Exponenten der Größe $n/2$.
- Ein derartiges Verfahren wird *Divide- und Conquer-Algorithmus* genannt.
- Rekursive Formulierung des Algorithmus:
  - $a^n \mod p = (a^{n/2} \mod p)^2 \mod p$ für gerade n
  - $a^n \mod p = (a^{n-1} \mod p) \cdot a \mod p$ für ungerade n
  - $a^0 \mod p = 1$

Inhaltsangabe

Verschlüsselungsverfahren

Verschlüsselungsverfahren

- Bei einem Verschlüsselungsverfahren wird eine Nachricht N mit Hilfe einer Funktion E und eines Schlüssels e verschlüsselt
$$K = E_e(N) .$$
- Die Dekodierung erfolgt mit der Funktion D und dem Schlüssel d:
$$N = D_d(K) = D_d(E_e(N)) .$$
- Die Funktionen E und D sollten effizient berechnet werden können.
- Ein Problem ist die Schlüsselübergabe.

Verschlüsselungsverfahren

Verschlüsselungsverfahren

- Bei einem Verschlüsselungsverfahren wird eine Nachricht N mit Hilfe einer Funktion E und eines Schlüssels e verschlüsselt
$$K = E_e(N) .$$
- Die Dekodierung erfolgt mit der Funktion D und dem Schlüssel d:
$$N = D_d(K) = D_d(E_e(N)) .$$
- Die Funktionen E und D sollten effizient berechnet werden können.
- Ein Problem ist die Schlüsselübergabe.

Verschlüsselungsverfahren

Verschlüsselungsverfahren

- Bei einem Verschlüsselungsverfahren wird eine Nachricht N mit Hilfe einer Funktion E und eines Schlüssels e verschlüsselt
$$K = E_e(N) .$$
- Die Dekodierung erfolgt mit der Funktion D und dem Schlüssel d:
$$N = D_d(K) = D_d(E_e(N)) .$$
- Die Funktionen E und D sollten effizient berechnet werden können.
- Ein Problem ist die Schlüsselübergabe.

Verschlüsselungsverfahren

Verschlüsselungsverfahren

- Bei einem Verschlüsselungsverfahren wird eine Nachricht N mit Hilfe einer Funktion E und eines Schlüssels e verschlüsselt
$$K = E_e(N) .$$
- Die Dekodierung erfolgt mit der Funktion D und dem Schlüssel d:
$$N = D_d(K) = D_d(E_e(N)) .$$
- Die Funktionen E und D sollten effizient berechnet werden können.
- Ein Problem ist die Schlüsselübergabe.

Asymmetrische Kryptographie

Asymmetrische Kryptographie

- Das RSA-Verfahren ist ein Beispiel eines *asymmetrischen* Verschlüsselungsverfahrens.
- Solche Verfahren wurden 1976 von Diffie und Hellman eingeführt.
- Hierbei verwenden Sender und Empfänger *jeweils eigene* Schlüssel e und d.
- Der Schlüssel e wird jeweils öffentlich bekannt gegeben, während der Schlüssel d geheim bleibt.
- Ein Schlüsselaustausch des persönlichen Dekodierungsschlüssels d ist demnach nicht erforderlich.

Asymmetrische Kryptographie

Asymmetrische Kryptographie

- Das RSA-Verfahren ist ein Beispiel eines *asymmetrischen* Verschlüsselungsverfahrens.
- Solche Verfahren wurden 1976 von Diffie und Hellman eingeführt.
- Hierbei verwenden Sender und Empfänger *jeweils eigene* Schlüssel e und d.
- Der Schlüssel e wird jeweils öffentlich bekannt gegeben, während der Schlüssel d geheim bleibt.
- Ein Schlüsselaustausch des persönlichen Dekodierungsschlüssels d ist demnach nicht erforderlich.

Asymmetrische Kryptographie

Asymmetrische Kryptographie

- Das RSA-Verfahren ist ein Beispiel eines *asymmetrischen* Verschlüsselungsverfahrens.
- Solche Verfahren wurden 1976 von Diffie und Hellman eingeführt.
- Hierbei verwenden Sender und Empfänger *jeweils eigene* Schlüssel e und d.
- Der Schlüssel e wird jeweils öffentlich bekannt gegeben, während der Schlüssel d geheim bleibt.
- Ein Schlüsselaustausch des persönlichen Dekodierungsschlüssels d ist demnach nicht erforderlich.

Asymmetrische Kryptographie

Asymmetrische Kryptographie

- Das RSA-Verfahren ist ein Beispiel eines *asymmetrischen* Verschlüsselungsverfahrens.
- Solche Verfahren wurden 1976 von Diffie und Hellman eingeführt.
- Hierbei verwenden Sender und Empfänger *jeweils eigene* Schlüssel e und d.
- Der Schlüssel e wird jeweils öffentlich bekannt gegeben, während der Schlüssel d geheim bleibt.
- Ein Schlüsselaustausch des persönlichen Dekodierungsschlüssels d ist demnach nicht erforderlich.

Asymmetrische Kryptographie

Asymmetrische Kryptographie

- Das RSA-Verfahren ist ein Beispiel eines *asymmetrischen* Verschlüsselungsverfahrens.
- Solche Verfahren wurden 1976 von Diffie und Hellman eingeführt.
- Hierbei verwenden Sender und Empfänger *jeweils eigene* Schlüssel e und d.
- Der Schlüssel e wird jeweils öffentlich bekannt gegeben, während der Schlüssel d geheim bleibt.
- Ein Schlüsselaustausch des persönlichen Dekodierungsschlüssels d ist demnach nicht erforderlich.

Das RSA-Verfahren

Kryptographisches Protokoll des RSA-Verfahrens (1978)

Der Empfänger und Teilnehmer beim RSA-Verfahren

- besorgt sich eine 400-stellige Dezimalzahl $m = p \cdot q$ mit 200-stelligen Primzahlen $p, q \in \mathbb{P}$,
- berechnet $\varphi = (p-1)(q-1)$,
- bestimmt und veröffentlicht einen öffentlichen Schlüssel e, der keinen gemeinsamen Teiler mit φ haben darf
- und berechnet seinen privaten Schlüssel d mit der Eigenschaft $e \cdot d = 1 \pmod{\varphi}$.
- Verschlüsselung und Entschlüsselung sind gegeben durch

$$K = E_e(N) = N^e \pmod{m} \quad \text{und} \quad D_d(K) = K^d \pmod{m}.$$

Das RSA-Verfahren

Kryptographisches Protokoll des RSA-Verfahrens (1978)

Der Empfänger und Teilnehmer beim RSA-Verfahren

- besorgt sich eine 400-stellige Dezimalzahl $m = p \cdot q$ mit 200-stelligen Primzahlen $p, q \in \mathbb{P}$,
- berechnet $\varphi = (p-1)(q-1)$,
- bestimmt und veröffentlicht einen öffentlichen Schlüssel e, der keinen gemeinsamen Teiler mit φ haben darf
- und berechnet seinen privaten Schlüssel d mit der Eigenschaft $e \cdot d = 1 \pmod{\varphi}$.
- Verschlüsselung und Entschlüsselung sind gegeben durch

$$K = E_e(N) = N^e \pmod{m} \quad \text{und} \quad D_d(K) = K^d \pmod{m}.$$

Das RSA-Verfahren

Kryptographisches Protokoll des RSA-Verfahrens (1978)

Der Empfänger und Teilnehmer beim RSA-Verfahren

- besorgt sich eine 400-stellige Dezimalzahl $m = p \cdot q$ mit 200-stelligen Primzahlen $p, q \in \mathbb{P}$,
- berechnet $\varphi = (p-1)(q-1)$,
- bestimmt und veröffentlicht einen öffentlichen Schlüssel e, der keinen gemeinsamen Teiler mit φ haben darf
- und berechnet seinen privaten Schlüssel d mit der Eigenschaft $e \cdot d = 1 \pmod{\varphi}$.
- Verschlüsselung und Entschlüsselung sind gegeben durch

$$K = E_e(N) = N^e \pmod{m} \quad \text{und} \quad D_d(K) = K^d \pmod{m}.$$

Das RSA-Verfahren

Kryptographisches Protokoll des RSA-Verfahrens (1978)

Der Empfänger und Teilnehmer beim RSA-Verfahren

- besorgt sich eine 400-stellige Dezimalzahl $m = p \cdot q$ mit 200-stelligen Primzahlen $p, q \in \mathbb{P}$,
- berechnet $\varphi = (p-1)(q-1)$,
- bestimmt und veröffentlicht einen öffentlichen Schlüssel e, der keinen gemeinsamen Teiler mit φ haben darf
- und berechnet seinen privaten Schlüssel d mit der Eigenschaft $e \cdot d = 1 \pmod{\varphi}$.
- Verschlüsselung und Entschlüsselung sind gegeben durch

$$K = E_e(N) = N^e \pmod{m} \quad \text{und} \quad D_d(K) = K^d \pmod{m}.$$

Das RSA-Verfahren

Kryptographisches Protokoll des RSA-Verfahrens (1978)

Der Empfänger und Teilnehmer beim RSA-Verfahren

- besorgt sich eine 400-stellige Dezimalzahl $m = p \cdot q$ mit 200-stelligen Primzahlen $p, q \in \mathbb{P}$,
- berechnet $\varphi = (p-1)(q-1)$,
- bestimmt und veröffentlicht einen öffentlichen Schlüssel e, der keinen gemeinsamen Teiler mit φ haben darf
- und berechnet seinen privaten Schlüssel d mit der Eigenschaft $e \cdot d = 1 \pmod{\varphi}$.
- Verschlüsselung und Entschlüsselung sind gegeben durch

$$K = E_e(N) = N^e \pmod{m} \quad \text{und} \quad D_d(K) = K^d \pmod{m}.$$

Das RSA-Verfahren

Kryptographisches Protokoll des RSA-Verfahrens (1978)

Der Empfänger und Teilnehmer beim RSA-Verfahren

- besorgt sich eine 400-stellige Dezimalzahl $m = p \cdot q$ mit 200-stelligen Primzahlen $p, q \in \mathbb{P}$,
- berechnet $\varphi = (p-1)(q-1)$,
- bestimmt und veröffentlicht einen öffentlichen Schlüssel e, der keinen gemeinsamen Teiler mit φ haben darf
- und berechnet seinen privaten Schlüssel d mit der Eigenschaft $e \cdot d = 1 \pmod{\varphi}$.
- Verschlüsselung und Entschlüsselung sind gegeben durch

$$K = E_e(N) = N^e \pmod{m} \quad \text{und} \quad D_d(K) = K^d \pmod{m}\,.$$

Das RSA-Verfahren

Was brauchen wir also für RSA?

- Bestimmung großer Primzahlen: `PrimeQ`
- Wir müssen möglichst effizient modulare Potenzen $N^e \pmod m$ berechnen: `PowerMod`
- Bestimmung des modularen Inversen $d = e^{-1} \pmod \varphi$: `PowerMod`
- Außerdem: Mit geeigneten Hilfsfunktionen wandeln wir unsere Nachrichten zuerst in Zahlen um und transformieren diese am Ende wieder zurück.

Das RSA-Verfahren

Was brauchen wir also für RSA?

- Bestimmung großer Primzahlen: `PrimeQ`
- Wir müssen möglichst effizient modulare Potenzen $N^e \pmod m$ berechnen: `PowerMod`
- Bestimmung des modularen Inversen $d = e^{-1} \pmod \varphi$: `PowerMod`
- Außerdem: Mit geeigneten Hilfsfunktionen wandeln wir unsere Nachrichten zuerst in Zahlen um und transformieren diese am Ende wieder zurück.

Das RSA-Verfahren

Was brauchen wir also für RSA?

- Bestimmung großer Primzahlen: `PrimeQ`
- Wir müssen möglichst effizient modulare Potenzen $N^e \pmod m$ berechnen: `PowerMod`
- Bestimmung des modularen Inversen $d = e^{-1} \pmod \varphi$: `PowerMod`
- Außerdem: Mit geeigneten Hilfsfunktionen wandeln wir unsere Nachrichten zuerst in Zahlen um und transformieren diese am Ende wieder zurück.

Das RSA-Verfahren

Was brauchen wir also für RSA?

- Bestimmung großer Primzahlen: `PrimeQ`
- Wir müssen möglichst effizient modulare Potenzen $N^e \pmod m$ berechnen: `PowerMod`
- Bestimmung des modularen Inversen $d = e^{-1} \pmod \varphi$: `PowerMod`
- Außerdem: Mit geeigneten Hilfsfunktionen wandeln wir unsere Nachrichten zuerst in Zahlen um und transformieren diese am Ende wieder zurück.

Finale

Erkenntnisse

Ich hoffe, mein Vortrag hat Ihnen Folgendes gezeigt:

- Mit geeigneten mathematischen Algorithmen kann man Geheimschriften so sicher machen, wie man möchte, so dass wirklich gilt: Geheim bleibt geheim!
- Also: Interneteinkauf, Online-Banking (mit `https`!), Computerzugriff von Ferne (`ssh`) etc. kann wirklich sicher sein.
- Mit Mathematik können wir lebenspraktische Probleme lösen. Und dies war nur ein Beispiel von vielen.
- Auch CD-Player, Handys, Satelliten, Navigationsgeräte und vieles mehr funktionieren nur mit viel „Mathe inside“!
- Mathematik macht Spaß!

Finale

Erkenntnisse

Ich hoffe, mein Vortrag hat Ihnen Folgendes gezeigt:

- Mit geeigneten mathematischen Algorithmen kann man Geheimschriften so sicher machen, wie man möchte, so dass wirklich gilt: Geheim bleibt geheim!
- Also: Interneteinkauf, Online-Banking (mit `https`!), Computerzugriff von Ferne (`ssh`) etc. kann wirklich sicher sein.
- Mit Mathematik können wir lebenspraktische Probleme lösen. Und dies war nur ein Beispiel von vielen.
- Auch CD-Player, Handys, Satelliten, Navigationsgeräte und vieles mehr funktionieren nur mit viel „Mathe inside“!
- Mathematik macht Spaß!

Finale

Erkenntnisse

Ich hoffe, mein Vortrag hat Ihnen Folgendes gezeigt:

- Mit geeigneten mathematischen Algorithmen kann man Geheimschriften so sicher machen, wie man möchte, so dass wirklich gilt: Geheim bleibt geheim!
- Also: Interneteinkauf, Online-Banking (mit `https`!), Computerzugriff von Ferne (`ssh`) etc. kann wirklich sicher sein.
- Mit Mathematik können wir lebenspraktische Probleme lösen. Und dies war nur ein Beispiel von vielen.
- Auch CD-Player, Handys, Satelliten, Navigationsgeräte und vieles mehr funktionieren nur mit viel „Mathe inside“!
- Mathematik macht Spaß!

Finale

Erkenntnisse

Ich hoffe, mein Vortrag hat Ihnen Folgendes gezeigt:

- Mit geeigneten mathematischen Algorithmen kann man Geheimschriften so sicher machen, wie man möchte, so dass wirklich gilt: Geheim bleibt geheim!
- Also: Interneteinkauf, Online-Banking (mit `https`!), Computerzugriff von Ferne (`ssh`) etc. kann wirklich sicher sein.
- Mit Mathematik können wir lebenspraktische Probleme lösen. Und dies war nur ein Beispiel von vielen.
- Auch CD-Player, Handys, Satelliten, Navigationsgeräte und vieles mehr funktionieren nur mit viel „Mathe inside“!
- Mathematik macht Spaß!

Finale

Erkenntnisse

Ich hoffe, mein Vortrag hat Ihnen Folgendes gezeigt:

- Mit geeigneten mathematischen Algorithmen kann man Geheimschriften so sicher machen, wie man möchte, so dass wirklich gilt: Geheim bleibt geheim!
- Also: Interneteinkauf, Online-Banking (mit `https`!), Computerzugriff von Ferne (`ssh`) etc. kann wirklich sicher sein.
- Mit Mathematik können wir lebenspraktische Probleme lösen. Und dies war nur ein Beispiel von vielen.
- Auch CD-Player, Handys, Satelliten, Navigationsgeräte und vieles mehr funktionieren nur mit viel „Mathe inside“!
- Mathematik macht Spaß!

Finale

Erkenntnisse

Ich hoffe, mein Vortrag hat Ihnen Folgendes gezeigt:

- Mit geeigneten mathematischen Algorithmen kann man Geheimschriften so sicher machen, wie man möchte, so dass wirklich gilt: Geheim bleibt geheim!
- Also: Interneteinkauf, Online-Banking (mit `https`!), Computerzugriff von Ferne (`ssh`) etc. kann wirklich sicher sein.
- Mit Mathematik können wir lebenspraktische Probleme lösen. Und dies war nur ein Beispiel von vielen.
- Auch CD-Player, Handys, Satelliten, Navigationsgeräte und vieles mehr funktionieren nur mit viel „Mathe inside“!
- Mathematik macht Spaß!

Vielen Dank für Ihr Interesse und guten Appetit in der Mensa!

Vielen Dank für Ihr Interesse und guten Appetit in der Mensa!