Public-Key-Algorithmus RSA

Karsten Heinze

Dresden, 18. Juli 2008

Gliederung

1 Motivation

Kennen Sie den? . . .

„Ich habe nichts zu verbergen.“

Kennen Sie den? . . .

„Ich habe nichts zu verbergen.“

Know-How, finanziele Situation, Kundenstamm,
Ausrüstung, Gesundheit, Personalbestand, Strategie,
Interessen, Forschung, politische Einstellung, . . .

Verschlüsselung

Symmetrische vs. asymmetrische Verfahren

- Symmetrische Verfahren
  - *ein* Schlüssel zum Ver- und Entschlüsseln
- Asymmetrische Verfahren (Public-Key-Verfahren)
  - ein *öffentlicher Schlüssel* zum Verschlüsseln
  - ein *geheimer Schlüssel* zum Entschlüsseln

Verschlüsselung

Symmetrische vs. asymmetrische Verfahren

- Symmetrische Verfahren
  - *ein* Schlüssel zum Ver- und Entschlüsseln
  - **Der Schlüssel muss *sicher* verteilt werden!**
- Asymmetrische Verfahren (Public-Key-Verfahren)
  - ein *öffentlicher Schlüssel* zum Verschlüsseln
  - ein *geheimer Schlüssel* zum Entschlüsseln

2 Algorithmus

Kurzbeschreibung

- populärster *Public-Key-Algorithmus*
- 1977 publiziert von *Rivest, Shamir* und *Adleman*
- geeignet für *Verschlüsselung* und *digitale Signaturen*
- US-Patent endete am 20. September 2000
- einfach zu verstehen

Funktionsweise

Ver-/Entschlüsseln eines Klar-/Geheimtextes

- Potentieren in einem *Restklassenring modulo n*
  - $Z_n = (M = \{0, 1, \ldots, n-1\}; +, *)$ mit $n = p * q$

Funktionsweise

Ver-/Entschlüsseln eines Klar-/Geheimtextes

- Potentieren in einem *Restklassenring modulo n*
  - $Z_n = (M = \{0, 1, \ldots, n-1\}; +, *)$ mit $n = p * q$
- öffentlicher Schlüssel: (e, n), privater Schlüssel: (d, n)

Funktionsweise

Ver-/Entschlüsseln eines Klar-/Geheimtextes

- Potentieren in einem *Restklassenring modulo n*
  - $Z_n = (M = \{0, 1, \ldots, n-1\}; +, *)$ mit $n = p * q$
- öffentlicher Schlüssel: (e, n), privater Schlüssel: (d, n)
- encrypt(m, e, n) : $m^e \equiv c \mod n$

  decrypt(c, d, n) : $c^d \equiv m \mod n$

Funktionsweise

Ver-/Entschlüsseln eines Klar-/Geheimtextes

- Potentieren in einem *Restklassenring modulo n*
  - $Z_n = (M = \{0, 1, \ldots, n-1\}; +, *)$ mit $n = p * q$
- öffentlicher Schlüssel: (e, n), privater Schlüssel: (d, n)
- encrypt(m, e, n) : $m^e \equiv c \mod n$

  decrypt(c, d, n) : $c^d \equiv m \mod n$

Wie hängen e und d zusammen ?

Funktionsweise

Ver-/Entschlüsseln eines Klar-/Geheimtextes

- Potentieren in einem *Restklassenring modulo n*
  - $Z_n = (M = \{0, 1, \ldots, n-1\}; +, *)$ mit $n = p * q$
- öffentlicher Schlüssel: (e, n), privater Schlüssel: (d, n)
- encrypt(m, e, n) : $m^e \equiv c \mod n$

  decrypt(c, d, n) : $c^d \equiv m \mod n$

Wie hängen e und d zusammen ?

$$\boxed{e = d^{-1}}$$

Multiplikatives Inverses

- existiert für alle zu n teilerfremden Elemente aus M

Multiplikatives Inverses

- existiert für alle zu n teilerfremden Elemente aus M

Wieviele Elemente aus M sind teilerfremd zu n?

Multiplikatives Inverses

- existiert für alle zu n teilerfremden Elemente aus M

Wieviele Elemente aus M sind teilerfremd zu n?

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

Euler'sche Funktion

Multiplikatives Inverses

- existiert für alle zu n teilerfremden Elemente aus M

Wieviele Elemente aus M sind teilerfremd zu n?

$$\phi(n) = (p-1) * (q-1)$$ *Euler'sche Funktion*

- aus dem *Euler-Fermat'schen Satz* folgt:

$$e * d \equiv 1 \mod \phi(n)$$

Multiplikatives Inverses berechnen

Mit Kenntnis von $\phi(n)$ trivial!

Multiplikatives Inverses berechnen

Mit Kenntnis von $\phi(n)$ trivial!

$$\boxed{e^{\phi(n)} \equiv 1 \mod n} \quad \textit{Euler-Fermat'scher Satz}$$

$$e^{\phi(n)-1} * e \equiv 1 \quad \rightarrow \quad e^{\phi(n)-1} \equiv e^{-1} \equiv \underline{\underline{d}} \mod n$$

Multiplikatives Inverses berechnen

Mit Kenntnis von $\phi(n)$ trivial!

$$\boxed{e^{\phi(n)} \equiv 1 \mod n} \quad \textit{Euler-Fermat'scher Satz}$$

$$e^{\phi(n)-1} * e \equiv 1 \quad \rightarrow \quad e^{\phi(n)-1} \equiv e^{-1} \equiv \underline{\underline{d}} \mod n$$

$|n| > 500$ Bit: Berechnung mit *Euklidischem Algorithmus*:

$$\mathrm{ggT}(e, \phi(n)) = (\alpha * e) + (\beta * \phi(n)) \quad \textit{Cofaktorendarst.}$$

Multiplikatives Inverses berechnen

Mit Kenntnis von $\phi(n)$ trivial!

$$\boxed{e^{\phi(n)} \equiv 1 \mod n} \quad \textit{Euler-Fermat'scher Satz}$$

$$e^{\phi(n)-1} * e \equiv 1 \quad \rightarrow \quad e^{\phi(n)-1} \equiv e^{-1} \equiv \underline{\underline{d}} \mod n$$

$|n| > 500$ Bit: Berechnung mit *Euklidischem Algorithmus*:

$$\begin{aligned} \mathrm{ggT}(e,\phi(n)) &= (\alpha * e) + (\beta * \phi(n)) && \textit{Cofaktorendarst.} \\ 1 &\equiv (\alpha * e) + \underbrace{(\beta * \phi(n))}_{0} && \mod \phi(n) \end{aligned}$$

Multiplikatives Inverses berechnen

Mit Kenntnis von $\phi(n)$ trivial!

$$\boxed{e^{\phi(n)} \equiv 1 \mod n} \quad \textit{Euler-Fermat'scher Satz}$$

$$e^{\phi(n)-1} * e \equiv 1 \quad \rightarrow \quad e^{\phi(n)-1} \equiv e^{-1} \equiv \underline{\underline{d}} \mod n$$

$|n| > 500$ Bit: Berechnung mit *Euklidischem Algorithmus*:

$$\begin{aligned}
\text{ggT}(e,\phi(n)) &= (\alpha * e) + (\beta * \phi(n)) && \textit{Cofaktorendarst.} \\
1 &\equiv (\alpha * e) + \underbrace{(\beta * \phi(n))}_{0} && \mod \phi(n) \\
1 &\equiv (\underline{\underline{d}} * e) && \mod \phi(n)
\end{aligned}$$

Sicherheit

- Wer $\phi(n)$ kennt, kann geh. Schlüssel berechnen!
  (Zu *jedem* öffentlichen Schlüssel bei Modul n!)
- Wer p und q kennt, kann $\phi(n)$ berechnen!
- Wer n faktorisieren kann, erhält p und q!

Sicherheit

- Wer $\phi(n)$ kennt, kann geh. Schlüssel berechnen!
  (Zu *jedem* öffentlichen Schlüssel bei Modul n!)
- Wer p und q kennt, kann $\phi(n)$ berechnen!
- Wer n faktorisieren kann, erhält p und q!

Das Faktorisieren großer Zahlen gilt als schwer!

Faktorisierungsannahme

- Primzahlmenge $P = \left\{ p \mid \forall q,\ 1 < q < p :\ q \nmid p \right\} \subset N$
- Nicht-Primzahlen *eindeutig* in Primfaktoren zerlegbar

Faktorisierungsannahme

- Primzahlmenge $P = \left\{ p \mid \forall q,\ 1 < q < p :\ q \nmid p \right\} \subset N$
- Nicht-Primzahlen *eindeutig* in Primfaktoren zerlegbar

Beispiele:

$$24 = 2^3 * 3$$

$$1.800 =$$

Faktorisierungsannahme

- Primzahlmenge $P = \left\{ p \mid \forall q,\ 1 < q < p :\ q \nmid p \right\} \subset N$
- Nicht-Primzahlen *eindeutig* in Primfaktoren zerlegbar

Beispiele:

$$24 = 2^3 * 3$$

$$1.800 = 2^3 * 3^2 * 5^2$$

$$122.960.200.847 =$$

Faktorisierungsannahme

- Primzahlmenge $P = \left\{ p \mid \forall q,\ 1 < q < p :\ q \nmid p \right\} \subset N$
- Nicht-Primzahlen *eindeutig* in Primfaktoren zerlegbar

Beispiele:

$$24 = 2^3 * 3$$

$$1.800 = 2^3 * 3^2 * 5^2$$

$$122.960.200.847 = 11^2 * 17^4 * 23^3$$

$$11.013.803.948.159 =$$

Faktorisierungsannahme

- Primzahlmenge $P = \left\{ p \mid \forall q,\ 1 < q < p :\ q \nmid p \right\} \subset N$
- Nicht-Primzahlen *eindeutig* in Primfaktoren zerlegbar

Beispiele:

$$
\begin{aligned}
24 &= 2^3 * 3 \\
1.800 &= 2^3 * 3^2 * 5^2 \\
122.960.200.847 &= 11^2 * 17^4 * 23^3 \\
11.013.803.948.159 &= 1.105.879 * 9.959.321
\end{aligned}
$$

Faktorisierung von RSA 640

$$
\begin{aligned}
\text{RSA}\,640 &= 3107418240490043721350750035888567930037346022842\\
&\quad 7275457201619488232064405180815045563468296717232\\
&\quad 8678243791627283803341547107310850191954852900733\\
&\quad 772482278352574238645401469173660247765234660 9\\
&= 1634733645809253848443133883865090859841783670033\\
&\quad 0923121811108523893331001045081512121 18167511579\\
&* 1900871281664822113126851573935413975471896789968\\
&\quad 515493666638539088027103802104498957191261465571
\end{aligned}
$$

Faktorisierung von RSA 640

$$
\begin{aligned}
\text{RSA}\,640 &= 3107418240490043721350750035888567930037346022842\\
&\quad 7275457201619488232064405180815045563468296717232\\
&\quad 8678243791627283803341547107310850191954852900733\\
&\quad 772482278352574238645401469173660247652346609\\
&= 1634733645809253848443133883865090859841783670033\\
&\quad 092312181110852389333100104508151212118167511579\\
&* 1900871281664822113126851573935413975471896789968\\
&\quad 515493666638539088027103802104498957191261465571
\end{aligned}
$$

- 80 Opteron CPUs (2.2-GHz) rechneten ca. 6 Monate
  Algorithmus: Allgemeines Zählkörpersieb (general number field sieve)
- in unmittelbarer Reichweite: 768 Bit RSA-Moduln

Was sollte man sich merken?

- RSA steht und fällt mit der Faktorisierungsannahme

Was sollte man sich merken?

- RSA steht und fällt mit der Faktorisierungsannahme
- „Haltbarkeit“ von Chiffraten und Signaturen begrenzt
  - *Moores Law* im Auge behalten
  - Modul n entsprechend groß wählen

Was sollte man sich merken?

- RSA steht und fällt mit der Faktorisierungsannahme
- „Haltbarkeit“ von Chiffraten und Signaturen begrenzt
  - *Moores Law* im Auge behalten
  - Modul n entsprechend groß wählen
- RSA ist langsam im Vergleich zu symm. Verfahren
  - in *hybriden Verfahren* einsetzen (Schlüsselverteilung)

3 Anwendung

Pretty Good Privacy (PGP)

Verschlüsseln und Signieren von E-Mails

- 1991: Version 1.0
- Phil Zimmermann, die NSA und die US-Gesetze . . .

Pretty Good Privacy (PGP)

Verschlüsseln und Signieren von E-Mails

- 1991: Version 1.0
- Phil Zimmermann, die NSA und die US-Gesetze . . .
- *Web of Trust* statt Hierarchie beglaubigter Keyserver

  (. . . die von Behörden, Geheimdiensten, Mafia etc. kontrolliert werden)

GNU Privacy Guard (GPG)

Vollständiger Ersatz für PGP (Open Source)

- 1997: Version 0.0.0 mit *elGamal* statt RSA
- läuft auf vielen Betriebssystemen (auch Windows!)
- Kommandozeilentool (grafische Frontends verfügbar)

GNU Privacy Guard (GPG)

Vollständiger Ersatz für PGP (Open Source)

- 1997: Version 0.0.0 mit *elGamal* statt RSA
- läuft auf vielen Betriebssystemen (auch Windows!)
- Kommandozeilentool (grafische Frontends verfügbar)

Sie nutzen *Thunderbird, Eudora, Evolution* oder *KMail*?

GNU Privacy Guard (GPG)

Vollständiger Ersatz für PGP (Open Source)

- 1997: Version 0.0.0 mit *elGamal* statt RSA
- läuft auf vielen Betriebssystemen (auch Windows!)
- Kommandozeilentool (grafische Frontends verfügbar)

Sie nutzen *Thunderbird, Eudora, Evolution* oder *KMail*?

Probieren Sie's doch mal aus ;)

Fragen?

Danke!

Anhang

Gruppentheorie

Restklassenringe modulo n $(n = p * q \mid p, q \in P)$

Gegeben: $M = \{0, 1, \ldots, n-1\}$

	$(M; +)$	$(M; *)$
Assoziativität	$(a+b)+c = a+(b+c)$	$(a*b)*c = a*(b*c)$
Kommutativität	$a+b = b+a$	$a*b = b*a$
neutrales Element	$a+e = a$	$a*e = a$
inverses Element	$a+a^{-1} = e$	— (nur, falls $a \not\mid n$)
	abelsche Gruppe	abelscher Monoid

$Z_n = (M; +, *)$ ist ein *Restklassenring modulo n*

Gruppentheorie

Euler'sche Funktion $\phi(m)$ und Euler-Fermat'scher Satz

- Wieviele Elemente aus M sind teilerfremd zu n?

$$\begin{aligned} \phi(n) &= \left| \left\{ a \;\middle|\; 1 \leq a < n \wedge \text{ggT}(a,n) = 1 \right\} \right| \\ &= (p-1) * (q-1) \end{aligned}$$

- für alle $\phi(n)$ teilerfremden Elemente gilt:
  - es existiert a^{-1}
  - $\boxed{a^{\phi(n)} \equiv 1 \bmod n}$ *Euler-Fermat'scher Satz*