Das RSA-Verfahren

Proseminar Kryptographische Protokolle SS 2009

Armin Litzel

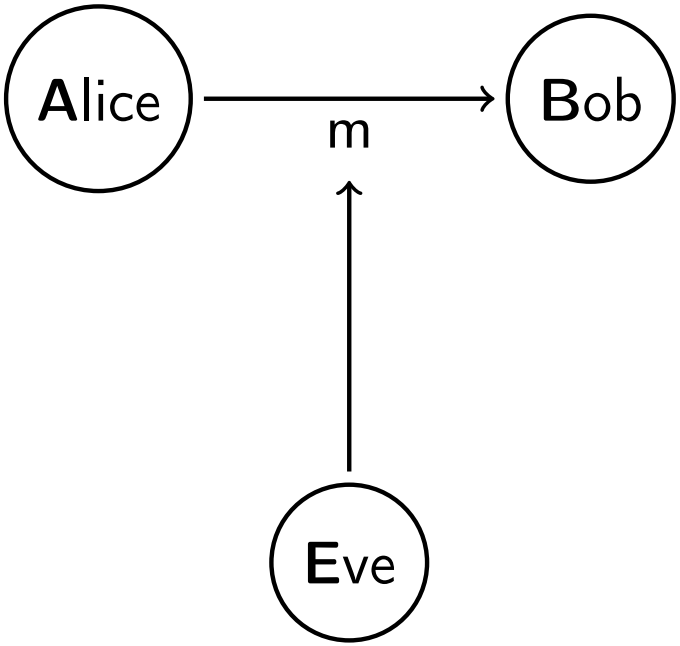
5.5.2009

Gliederung

Über das RSA Verfahren

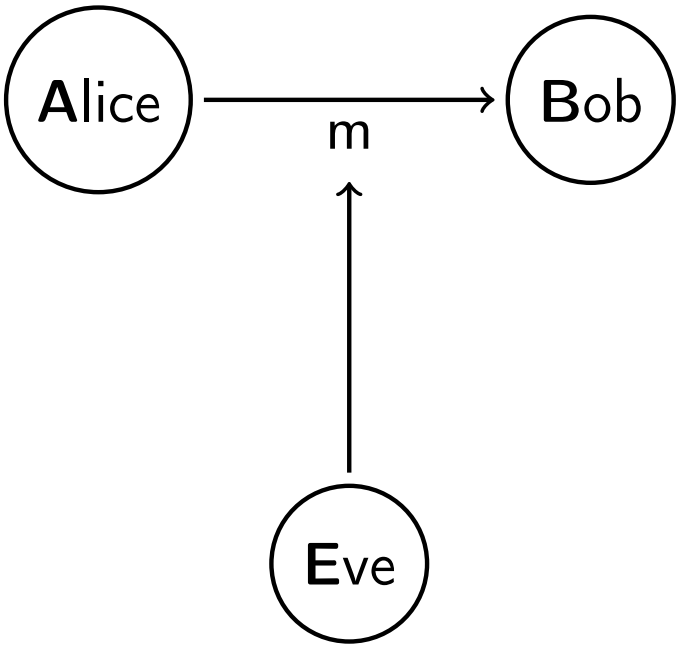
- 1977 entwickelt von Ronald L. **R**ivest, Adi **S**hamir und Leonard **A**dleman
- Erstes asymmetrisches Verschlüsselungsverfahren

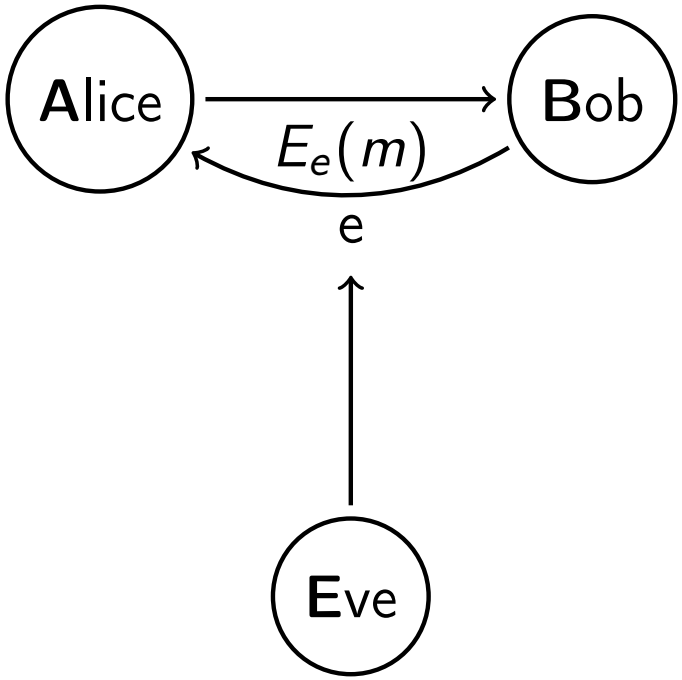
Ausgangssituation: Nachricht von A nach B



- Nachricht m von Alice an Bob
- Eve versucht diese Nachricht abzufangen

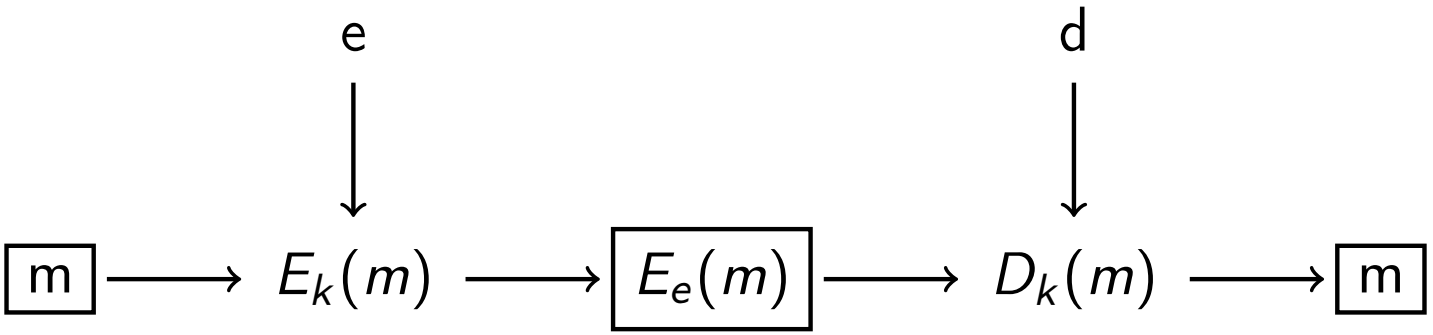
Asymmetrische Verschlüsselung





- Bei asymmetrischer Verschlüsselung kann Eve nur den public-key e abfangen
- Eve kann deshalb nicht die Nachricht entschlüsseln

Asymmetrische Verschlüsselung



- $e \neq d \leftrightarrow$ symmetrische Verschlüsselung: $e = d$
- $d = f(e)$
- $f^{-1}(e)$ nicht effizient berechenbar

Gliederung

Vereinfachte Darstellung des RSA-Verfahrens

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

1. Bob wählt zwei Primzahlen p und$q$, sodass $n = p \cdot q$, mit $n > m$.
2. Bob wählt eine Primzahl e mit $1 < e < \varphi(n) = (p-1) \cdot (q-1)$. Er bestimmt eine Zahl $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$, sodass $(e \cdot d) \mod \varphi(n) = 1$
3. Bob teilt Alice seinen öffentlichen Schlüssel (n, e) mit.

1. Sei $m = 3, p = 3$, $q = 11 => n = 33$. Es gilt $n > m$.
2. $\varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$. Sei $e := 7$. Es gilt $1 < e < \varphi(n)$. Für $d := 3$ gilt $(e \cdot d) \mod \varphi(n) = 1$, da $21 \mod 20 = 1$.
3. Der öffentliche Schlüssel ist $(33, 7)$.

Vereinfachte Darstellung des RSA-Verfahrens

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

1. Bob wählt zwei Primzahlen p und$q$, sodass $n = p \cdot q$, mit $n > m$.
2. Bob wählt eine Primzahl e mit $1 < e < \varphi(n) = (p-1) \cdot (q-1)$. Er bestimmt eine Zahl $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$, sodass $(e \cdot d) \mod \varphi(n) = 1$
3. Bob teilt Alice seinen öffentlichen Schlüssel (n, e) mit.

1. Sei $m = 3, p = 3$, $q = 11 => n = 33$. Es gilt $n > m$.
2. $\varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$. Sei $e := 7$. Es gilt $1 < e < \varphi(n)$. Für $d := 3$ gilt $(e \cdot d) \mod \varphi(n) = 1$, da $21 \mod 20 = 1$.
3. Der öffentliche Schlüssel ist $(33, 7)$.

Vereinfachte Darstellung des RSA-Verfahrens

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

1. Bob wählt zwei Primzahlen p und$q$, sodass $n = p \cdot q$, mit $n > m$.
2. Bob wählt eine Primzahl e mit $1 < e < \varphi(n) = (p-1) \cdot (q-1)$. Er bestimmt eine Zahl $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$, sodass $(e \cdot d) \mod \varphi(n) = 1$
3. Bob teilt Alice seinen öffentlichen Schlüssel (n, e) mit.

1. Sei $m = 3, p = 3$, $q = 11 => n = 33$. Es gilt $n > m$.
2. $\varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$. Sei $e := 7$. Es gilt $1 < e < \varphi(n)$. Für $d := 3$ gilt $(e \cdot d) \mod \varphi(n) = 1$, da $21 \mod 20 = 1$.
3. Der öffentliche Schlüssel ist $(33, 7)$.

Vereinfachte Darstellung des RSA-Verfahrens

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

1. Bob wählt zwei Primzahlen p und$q$, sodass $n = p \cdot q$, mit $n > m$.
2. Bob wählt eine Primzahl e mit $1 < e < \varphi(n) = (p-1) \cdot (q-1)$. Er bestimmt eine Zahl $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$, sodass $(e \cdot d) \mod \varphi(n) = 1$
3. Bob teilt Alice seinen öffentlichen Schlüssel (n, e) mit.

1. Sei $m = 3, p = 3$, $q = 11 => n = 33$. Es gilt $n > m$.
2. $\varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$. Sei $e := 7$. Es gilt $1 < e < \varphi(n)$. Für $d := 3$ gilt $(e \cdot d) \mod \varphi(n) = 1$, da $21 \mod 20 = 1$.
3. Der öffentliche Schlüssel ist $(33, 7)$.

Vereinfachte Darstellung des RSA-Verfahrens

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

1. Bob wählt zwei Primzahlen p und$q$, sodass $n = p \cdot q$, mit $n > m$.
2. Bob wählt eine Primzahl e mit $1 < e < \varphi(n) = (p-1) \cdot (q-1)$. Er bestimmt eine Zahl $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$, sodass $(e \cdot d) \mod \varphi(n) = 1$
3. Bob teilt Alice seinen öffentlichen Schlüssel (n, e) mit.

1. Sei $m = 3, p = 3$, $q = 11 => n = 33$. Es gilt $n > m$.
2. $\varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$. Sei $e := 7$. Es gilt $1 < e < \varphi(n)$. Für $d := 3$ gilt $(e \cdot d) \mod \varphi(n) = 1$, da $21 \mod 20 = 1$.
3. Der öffentliche Schlüssel ist $(33, 7)$.

Vereinfachte Darstellung des RSA-Verfahrens

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

1. Bob wählt zwei Primzahlen p und$q$, sodass $n = p \cdot q$, mit $n > m$.
2. Bob wählt eine Primzahl e mit $1 < e < \varphi(n) = (p-1) \cdot (q-1)$. Er bestimmt eine Zahl $d \in \mathbb{N}$ mit $1 < d < \varphi(n)$, sodass $(e \cdot d) \mod \varphi(n) = 1$
3. Bob teilt Alice seinen öffentlichen Schlüssel (n, e) mit.

1. Sei $m = 3, p = 3$, $q = 11 => n = 33$. Es gilt $n > m$.
2. $\varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$. Sei $e := 7$. Es gilt $1 < e < \varphi(n)$. Für $d := 3$ gilt $(e \cdot d) \mod \varphi(n) = 1$, da $21 \mod 20 = 1$.
3. Der öffentliche Schlüssel ist $(33, 7)$.

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

5. Alice berechnet den Geheimtext $c = E(m) = m^e \mod n$ und schickt ihn an Bob.
6. Bob berechnet den Klartext $D(c) = c^d \mod n = m$.

5. Der Geheimtext ist $3^7 \mod 33 = 2187 \mod 33 = 2187 - 33 \cdot 66 = 2187 - 2178 = 9$.
6. Der Klartext ist $9^3 \mod 33 = 729 \mod 33 = 729 - 33 \cdot 22 = 729 - 726 = 3 \mod 33 = 3$.

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

5. Alice berechnet den Geheimtext $c = E(m) = m^e \mod n$ und schickt ihn an Bob.
6. Bob berechnet den Klartext $D(c) = c^d \mod n = m$.

5. Der Geheimtext ist $3^7 \mod 33 = 2187 \mod 33 = 2187 - 33 \cdot 66 = 2187 - 2178 = 9$.
6. Der Klartext ist $9^3 \mod 33 = 729 \mod 33 = 729 - 33 \cdot 22 = 729 - 726 = 3 \mod 33 = 3$.

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

5. Alice berechnet den Geheimtext $c = E(m) = m^e \mod n$ und schickt ihn an Bob.
6. Bob berechnet den Klartext $D(c) = c^d \mod n = m$.

5. Der Geheimtext ist $3^7 \mod 33 = 2187 \mod 33 = 2187 - 33 \cdot 66 = 2187 - 2178 = 9$.
6. Der Klartext ist $9^3 \mod 33 = 729 \mod 33 = 729 - 33 \cdot 22 = 729 - 726 = 3 \mod 33 = 3$.

Verschlüsselung einer Nachricht $M \in \mathbb{N}$

5. Alice berechnet den Geheimtext $c = E(m) = m^e \mod n$ und schickt ihn an Bob.
6. Bob berechnet den Klartext $D(c) = c^d \mod n = m$.

5. Der Geheimtext ist $3^7 \mod 33 = 2187 \mod 33 = 2187 - 33 \cdot 66 = 2187 - 2178 = 9$.
6. Der Klartext ist $9^3 \mod 33 = 729 \mod 33 = 729 - 33 \cdot 22 = 729 - 726 = 3 \mod 33 = 3$.

Gliederung

Digitale Signaturen mit RSA

- Digitale Signaturen sind vergleichbar mit Unterschriften.
- Mit ihnen kann man verifizieren, welche Person ein Dokument signiert hat.

Signieren von Dokumenten mit RSA

Alice will das Dokument $m \in \mathbb{N}$ mit $m < n$ signieren.

1. Alice berechnet mit dem privaten Schlüssel (n,d) die Signatur $s = m^d \mod n$.
2. Bob berechnet mit dem öffentlichen Schlüssel (n,e) von Alice $s^e \mod n$.
3. Falls $s^e \mod n = m$ ist, dann weiß Bob, dass Alice m signiert hat.

Digitale Signaturen mit RSA

- Digitale Signaturen sind vergleichbar mit Unterschriften.
- Mit ihnen kann man verifizieren, welche Person ein Dokument signiert hat.

Signieren von Dokumenten mit RSA

Alice will das Dokument $m \in \mathbb{N}$ mit $m < n$ signieren.

1. Alice berechnet mit dem privaten Schlüssel (n,d) die Signatur $s = m^d \mod n$.
2. Bob berechnet mit dem öffentlichen Schlüssel (n,e) von Alice $s^e \mod n$.
3. Falls $s^e \mod n = m$ ist, dann weiß Bob, dass Alice m signiert hat.

Gliederung

Zahlentheoretische Grundlagen

Zur Erinnerung:

Restklassenring

- Für $n \in \mathbb{N}$ existieren n Restklassen modulo n: $[0, \ldots, n-1]$
- Die Restklassen bilden einen kommutativen Ring

Deshalb gilt:

1. $a \mod n \cdot b \mod n \equiv (a \cdot b) \mod n$
2. $(a^e) \equiv (a \mod n)^e \mod n$ für $e \in \mathbb{N}$

Das RSA-Theorem

Satz

Seien p und q zwei verschiedene Primzahlen und wird (e, d) so gewählt, dass

- $e \in [0, ..., \phi(pq) - 1]$ *mit* $ggT(e, \phi(pq)) = 1$
- $d \in [0, ..., \phi(pq) - 1]$ *mit* $e \cdot d \equiv 1 \mod \phi(pq)$

dann gilt für $m \in [0, ..., \phi(pq) - 1]$*:* $m^{ed} \equiv m \mod pq$

Beweis

Beweis

1. Es gilt $ed \mod \varphi(pq) = 1$. Deswegen gibt es eine ganze Zahl l mit $0 \leq l < \varphi(pq)$, sodass $ed = l \cdot \varphi(pq) + 1$ (Teilbarkeit).
2. Daraus folgt: $m^{ed} \mod pq = m^{l \cdot \varphi(n)+1} \mod pq$.

Satz

Seien p und q zwei verschiedene Primzahlen. Aus $a \equiv b \mod p$ und $a \equiv b \mod q$ folgt $a \equiv b \mod pq$.

3. Entweder p ist Teiler von m oder p ist kein Teiler von m. Dasselbe gilt für q.

Beweis

Beweis

1. Es gilt $ed \mod \varphi(pq) = 1$. Deswegen gibt es eine ganze Zahl l mit $0 \leq l < \varphi(pq)$, sodass $ed = l \cdot \varphi(pq) + 1$ (Teilbarkeit).
2. Daraus folgt: $m^{ed} \mod pq = m^{l \cdot \varphi(n)+1} \mod pq$.

Satz

Seien p und q zwei verschiedene Primzahlen. Aus $a \equiv b \mod p$ und $a \equiv b \mod q$ folgt $a \equiv b \mod pq$.

3. Entweder p ist Teiler von m oder p ist kein Teiler von m. Dasselbe gilt für q.

Beweis

Beweis

1. Es gilt $ed \mod \varphi(pq) = 1$. Deswegen gibt es eine ganze Zahl l mit $0 \leq l < \varphi(pq)$, sodass $ed = l \cdot \varphi(pq) + 1$ (Teilbarkeit).
2. Daraus folgt: $m^{ed} \mod pq = m^{l \cdot \varphi(n)+1} \mod pq$.

Satz

Seien p und q zwei verschiedene Primzahlen. Aus $a \equiv b \mod p$ und $a \equiv b \mod q$ folgt $a \equiv b \mod pq$.

3. Entweder p ist Teiler von m oder p ist kein Teiler von m. Dasselbe gilt für q.

Beweis Fortsetzung

Wenn p Teiler von m ist, dann gilt $m \mod p = 0$.

$$\begin{aligned} m^{ed} \mod p &= (m \cdot m^{l \cdot \varphi(pq)}) \mod p \\ &= (m \mod p \cdot m^{l \cdot \varphi(pq)} \mod p) \mod p \\ &= (0 \cdot m^{l \cdot \varphi(pq)} \mod p) \mod p \\ &= 0 \\ &= m \mod p \end{aligned}$$

Beweis Fortsetzung

Wenn p kein Teiler von m ist:

Kleiner Satz von Fermat

Für $m \in \mathbb{N}$ und eine Primzahl p mit $ggT(p, m) = 1$ gilt: $m^{p-1} \mod p = 1$

$$\begin{aligned} m^{ed} &= (m \mod p \cdot (m^{p-1})^{l(q-1)} \mod p) \mod p \\ &= (m \mod p \cdot (m^{p-1} \mod p)^{l\cdot(q-1)} \mod p) \mod p \\ &= m \mod p \end{aligned}$$

Analog kann für q gezeigt werden: $m^{ed} \mod q = m \mod q$.
Da $m^{ed} \equiv m \mod p$ und $m^{ed} \equiv m \mod q$ ist die Behauptung $m^{ed} \equiv m \mod pq$ bewiesen.

Gliederung

Angriff durch Faktorisierung des Moduls n

- Da der öffentliche Schlüssel (e, n) bekannt ist, wäre die Sicherheit von RSA nicht gewährleistet wenn p und q aus n ermittelt werden könnten. Warum?
  - Aus der Kenntnis von p und q lässt sich auch $\varphi(n)$ berechnen.
  - Mit $(e \cdot d) \equiv 1 \mod \varphi(n)$ kann schließlich der öffentliche Schlüssel (d, n) bestimmt werden.

Angriff durch Faktorisierung des Moduls n

- Da der öffentliche Schlüssel (e, n) bekannt ist, wäre die Sicherheit von RSA nicht gewährleistet wenn p und q aus n ermittelt werden könnten. Warum?
  - Aus der Kenntnis von p und q lässt sich auch $\varphi(n)$ berechnen.
  - Mit $(e \cdot d) \equiv 1 \mod \varphi(n)$ kann schließlich der öffentliche Schlüssel (d, n) bestimmt werden.

Angriff durch Faktorisierung des Moduls n

- Da der öffentliche Schlüssel (e, n) bekannt ist, wäre die Sicherheit von RSA nicht gewährleistet wenn p und q aus n ermittelt werden könnten. Warum?
  - Aus der Kenntnis von p und q lässt sich auch $\varphi(n)$ berechnen.
  - Mit $(e \cdot d) \equiv 1 \mod \varphi(n)$ kann schließlich der öffentliche Schlüssel (d, n) bestimmt werden.

Sicherheit vor Faktorisierung

- Da kein Algorithmus bekannt ist, der eine große Zahl n effizient in ihre Primfaktoren faktorisieren kann, ist die Sicherheit der Schlüssel bei RSA gewährleistet.
- Klassische Algorithmen: $\mathcal{O}(e^{\sqrt{n}})$.
- Shor-Algorithmus: $\mathcal{O}((\log n)^3)$.
- Je größer n ist, umso aufwendiger ist die Faktorisierung in ihre Primfaktoren.
- Wenn sich die Primfaktoren in ihrer Länge unterscheiden, ist es für bisher bekannte Algorithmen aufwendiger die Zahl n zu faktorieren.

Sicherheit vor Faktorisierung

- Da kein Algorithmus bekannt ist, der eine große Zahl n effizient in ihre Primfaktoren faktorisieren kann, ist die Sicherheit der Schlüssel bei RSA gewährleistet.
- Klassische Algorithmen: $\mathcal{O}(e^{\sqrt{n}})$.
- Shor-Algorithmus: $\mathcal{O}((\log n)^3)$.
- Je größer n ist, umso aufwendiger ist die Faktorisierung in ihre Primfaktoren.
- Wenn sich die Primfaktoren in ihrer Länge unterscheiden, ist es für bisher bekannte Algorithmen aufwendiger die Zahl n zu faktorieren.

The RSA Factoring Challenge

- RSA Security Inc. setzte regelmäßig Prämien zur Faktorisierung von großen Zahlen aus.
- Die zuletzt faktorisierte Zahl (640 Bit) bestand aus folgenden Primfaktoren:

$$163473364580925384844313388386509085984178367003309231218111085238933310010450815121211816751157 9$$

und

$$190087128166482211312685157393541397547189678996851549366663853908802710380210449895719126146557 1$$

The RSA Factoring Challenge

- Eine Forschergruppe des BSI gewann den Wettbewerb mit 20.000 US-Dollar Preisgeld.
- Es wurden 20 Computer mit 2,2 Ghz AMD Opteron Prozessoren eingesetzt,
- Die Primfaktorzerlegung benötigte etwa fünf Monate Rechenzeit.
- Im Mai 2007 wurde außerhalb der RSA-Factoring-Challenge eine 1039-bit Zahl faktorisiert
- Seitdem wurde der Wettbewerb eingestellt.

The RSA Factoring Challenge

- Eine Forschergruppe des BSI gewann den Wettbewerb mit 20.000 US-Dollar Preisgeld.
- Es wurden 20 Computer mit 2,2 Ghz AMD Opteron Prozessoren eingesetzt,
- Die Primfaktorzerlegung benötigte etwa fünf Monate Rechenzeit.
- Im Mai 2007 wurde außerhalb der RSA-Factoring-Challenge eine 1039-bit Zahl faktorisiert
- Seitdem wurde der Wettbewerb eingestellt.

Gliederung

Implementierung von RSA

- Es ist notwendig eine BigInt-Bibliothek zu verwenden
- Zur Primzahlermittlung eignen sich Algorithmen auf dem Satz von Fermat basieren. Sehr effizient ist das Miller-Rabin-Primtest.

- Texte werden mit Zeichensätzen (ASCII) in Ziffernblöcke umgewandelt
- Es ist sinvoll die Blocklänge auf 32- oder 64-Bit festzulegen
  - 32-Bit Register: | 8bit | 8bit | 8bit | 8bit |
  - 64-Bit Register: | 8bit | 8bit | 8bit | 8bit | 8bit | 8bit | 8bit | 8bit |

Implementierung von RSA

- Es ist notwendig eine BigInt-Bibliothek zu verwenden
- Zur Primzahlermittlung eignen sich Algorithmen auf dem Satz von Fermat basieren. Sehr effizient ist das Miller-Rabin-Primtest.

- Texte werden mit Zeichensätzen (ASCII) in Ziffernblöcke umgewandelt
- Es ist sinvoll die Blocklänge auf 32- oder 64-Bit festzulegen
  - 32-Bit Register:

8bit	8bit	8bit	8bit

  - 64-Bit Register:

8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit

Schlüsselerzeugung

- Mit dem erweiterten euklidischen Algorithmus kann d aus e und $\varphi(n)$ berechnet werden:
- Wegen $ggT(\varphi(n), e) = 1$ gibt es ein e und d mit $1 = e \cdot d + l\varphi(n)$ (Die kleinste positive Linearkombination ergibt den ggT).
- Der erweiterte euklidische Algorithmus berechnet neben dem ggT die Koeffizienten der Linearfaktordarstellung .

Der Euklidische Algorithmus

Euklidischer Algorithmus

Für ganze Zahlen $a > 0$ und $b \geq 0$ gilt: $ggT(a,b) = ggT(b, a \mod b)$. Mit $ggT(a,0) = a$ hat man den ggt von a und b gefunden.

Euklidischer Algorithmus

$ggT(1071,1029) = ggT(1029,1071 \mod 1029) = ggT(1029,42) = ggT(42,1029 \mod 42) = ggT(42,21) = ggT(21,42 \mod 21) = ggT(21,0) = 21$

Erweiterter Euklidischer Algorithmus

Sei $r_0 = a, r_1 = b$ sowie $x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1$

Sei $x_{k+1} = q_k x_k + x_{k-1}$, $y_{k+1} = q_k y_k + y_{k-1}$ mit $q_k = r_{k-1} divr_k$

Dann gilt $r_n = (-1)^n x_n a + (-1)^{n+1} y_n b$

Der Euklidische Algorithmus

Euklidischer Algorithmus

Für ganze Zahlen $a > 0$ und $b \geq 0$ gilt: $ggT(a,b) = ggT(b, a \mod b)$. Mit $ggT(a,0) = a$ hat man den ggt von a und b gefunden.

Euklidischer Algorithmus

$ggT(1071,1029) = ggT(1029,1071 \mod 1029) = ggT(1029,42) = ggT(42,1029 \mod 42) = ggT(42,21) = ggT(21,42 \mod 21) = ggT(21,0) = 21$

Erweiterter Euklidischer Algorithmus

Sei $r_0 = a, r_1 = b$ sowie $x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1$
Sei $x_{k+1} = q_k x_k + x_{k-1}$, $y_{k+1} = q_k y_k + y_{k-1}$ mit $q_k = r_{k-1} divr_k$
Dann gilt $r_n = (-1)^n x_n a + (-1)^{n+1} y_n b$

Erweiterter Euklidischer Algorithmus zur Schlüsselerzeugung

Erweiterter Euklidischer Algorithmus für $e = 7$, $\phi(n) = 20$

k	0	1
r_k	20	7
q_k		2
x_k	1	0
y_k	0	1

Erweiterter Euklidischer Algorithmus zur Schlüsselerzeugung

Erweiterter Euklidischer Algorithmus für $e = 7$, $\phi(n) = 20$

k	0	1	2
r_k	20	7	6
q_k		2	1
x_k	1	0	1
y_k	0	1	2

- $x_k = q_{k-1} \cdot x_{k-1} + x_{k-2}$, $y_k = q_{k-1} \cdot y_{k-1} + y_{k-2}$

Erweiterter Euklidischer Algorithmus zur Schlüsselerzeugung

Erweiterter Euklidischer Algorithmus für $e = 7$, $\phi(n) = 20$

k	0	1	2	3	4
r_k	20	7	6	1	0
q_k		2	1	6	0
x_k	1	0	1	1	7
y_k	0	1	2	3	20

- $r_3 = 1 = ggt(\varphi(n), e) => n = 3$
- $1 = (-1)^3 x_3 \varphi(n) + (-1)^4 y_3 e = (-1) \cdot 20 + 3 \cdot 7$
- Vermutung d=3
- Probe $(e \cdot d) \mod \varphi(n) = 1$?
- $(21 \mod 20) = 1$

Erweiterter Euklidischer Algorithmus zur Schlüsselerzeugung

Erweiterter Euklidischer Algorithmus für $e = 7$, $\phi(n) = 20$

k	0	1	2	3	4
r_k	20	7	6	1	0
q_k		2	1	6	0
x_k	1	0	1	1	7
y_k	0	1	2	3	20

- $r_3 = 1 = ggt(\varphi(n), e) => n = 3$
- $1 = (-1)^3 x_3 \varphi(n) + (-1)^4 y_3 e = (-1) \cdot 20 + 3 \cdot 7$
- Vermutung d=3
- Probe $(e \cdot d) \mod \varphi(n) = 1$?
- $(21 \mod 20) = 1$

Zusammenfassung

- RSA ist ein asymmetrisches Verschlüsselungsverfahren, das auf zahlentheoretischen Problemen beruht.
- Die Sicherheit ist gewährleistet, wenn $n \geq 2048 Bit$. p und q sollten sich auch in der Länge unterscheiden.

- Ausblick
  - Es wird immer schnellere Computer geben und effizientere Algorithmen zur Berechnung der Primfaktorzerlegung.
  - Das RSA-Verfahren wird deshalb in Zukunft von anderen Verfahren verdrängt werden, die mit einer niedrigeren Schlüssellänge auskommen.

Weiterführende Literatur I


Claudia Eckert
IT-Sicherheit: Konzepte, Verfahren, Protokolle.
5.überarbeitete Auflage, R. Oldenbourg-Verlag, 2007.


Johannes Buchmann.
Einführung in die Kryptographie.
3. erweiterte Auflage. Springer, Berlin, 2004.