Public-Key-Kryptographie
mit dem RSA-Schema

Andreas Meisel und Robert Mileski

Institut für Informatik der Universität Potsdam
Seminar „Kryptographie und Datensicherheit“
WS 2006/2007

Inhaltsverzeichnis

Geschichte der Kryptographie (1/3)

- 1900 v. Chr.: Erste Anwendungen von kryptographischen Umformungen bei Grabinschriften; älteste bekannte Anwendung der Kryptographie
- 600-500 v. Chr.: hebräische Gelehrte benutzten einfache Zeichenaustauschalgorithmen
- 475 v. Chr.: Sparta entwickelt das erste Verschlüsselungsgerät: die Skytale
- 350 v. Chr.: Erstes Buch über Kryptographie vom griechischen Historiker Aeneas, dem Taktiker
- 60 v. Chr. Verwendung von Substitutionschiffren durch Julius Caesar
- 1412: älteste bekannte Abhandlung über Kryptoanalyse vom ägyptischen Gelehrten al-Kalkashandi veröffentlicht

Geschichte der Kryptographie (2/3)

- Ende des 19.Jahrhunderts: Kerkhoffs-Prinzip
- 1917: Edward Hugh Hebern erfindet die erste Rotormaschine
- 1918: Patentierung der ENIGMA von Scherbius
- 1971: IBM erfindet das Lucifer encryption scheme
- 1975: Data Encryption Standard (DES)
- Anfang der 1970er Jahre: im Government Communications Headquarters von Ellis, Cocks und Williamson entwickeltes asymmetrisches Verfahren
- 1976: Erste öffentliche Präsentation des Public-Key-Konzepts von Diffie und Hellman
- 1980: Patentierung von RSA (Rivest, Shamir und Adleman)

Geschichte der Kryptographie (3/3)

- 1991 Phil Zimmermann schrieb die erste Version von PGP (Pretty Good Privacy)
- 21. September 2000: Patent zu RSA ist abgelaufen

Definitionen (1/3)

Kryptosystem:

„Ein Kryptosystem ist ein Verfahren, bei dem eine Eingabemenge (Klartext), gesteuert durch Parameter (Schlüssel), in eine Ausgabemenge (Geheimtext) gewandelt werden kann, und umgekehrt, der Geheimtext wieder in den Klartext zurückgewandelt werden kann.
Auf diese Weise lassen sich Informationen vor unbefugtem Zugriff schützen und den Inhalt der Botschaft nur dem gewünschten Empfänger zugänglich machen.“

Quelle: http://de.wikipedia.org/wiki/Kryptosystem

Definitionen (2/3)

Symmetrische Kryptosysteme:

Bei symmetrischen Kryptosystemen sind die Schlüssel zum Ver- und Entschlüsseln einer Nachricht identisch.

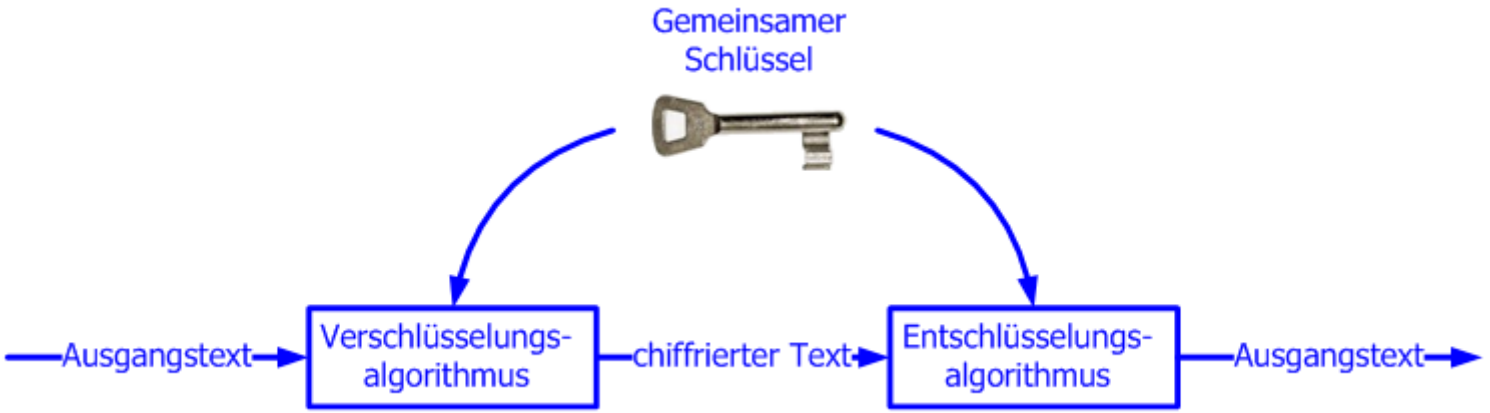


Abbildung 1: Vereinfachtes Modell der symmetrischen Verschlüsselung

Definitionen (3/3)

Asymmetrische Kryptosysteme:

Bei asymmetrischen Kryptosystemen unterscheiden sich die Schlüssel zum Ver- und Entschlüsseln einer Nachricht. Diese Verfahren werden auch als Public-Key-Verfahren bezeichnet.

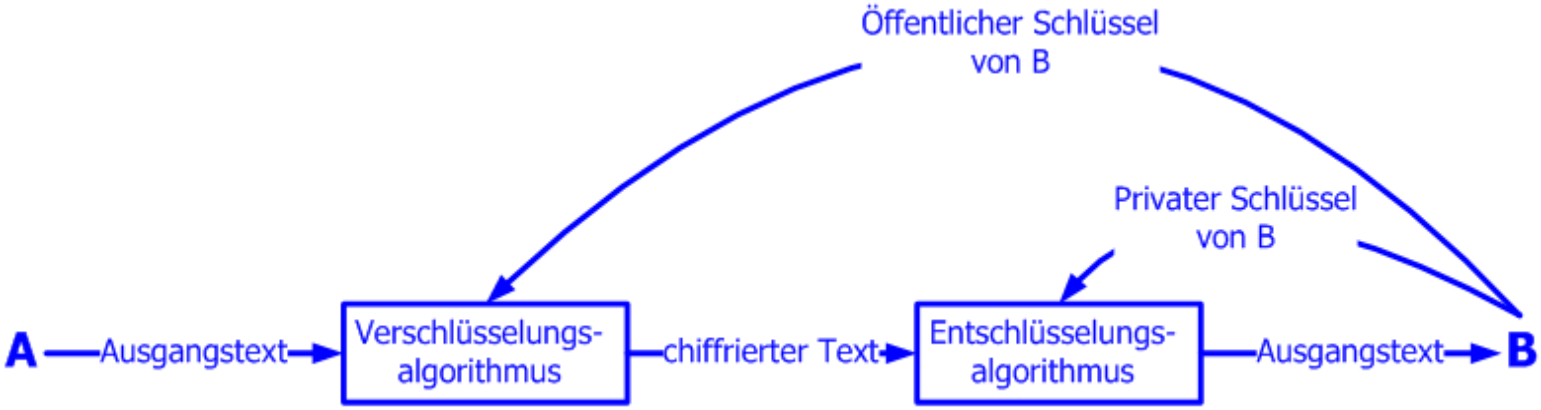


Abbildung 2: Vereinfachtes Modell der Public-Key-Verschlüsselung

Schlüsselvergabe (1/2)

- **Symmetrische Kryptosysteme:**

1. Ein Schlüssel kann von A gewählt und B physisch übergeben werden.
2. Eine dritte Partei kann den Schlüssel auswählen und ihn physisch an A und B ausliefern.
3. Wenn A und B kurz zuvor einen Schlüssel verwendet haben, kann eine Partei den neuen Schlüssel an die andere übertragen, der mit Hilfe des alten Schlüssel verschlüsselt wurde.
4. Wenn A und B beide über eine verschlüsselte Verbindung zu einer dritten Partei C verfügen, kann C über die verschlüsselten Verbindungen einen Schlüssel an A und B übertragen

Schlüsselvergabe (2/2)

- **Public-Key-Kryptosysteme:**
  - Öffentlicher Schlüssel:
    - Öffentliche Bekanntgabe
    - Öffentlich zugängliches Verzeichnis
    - Public-Key-Verwaltung
    - Public-Key-Zertifikate

Vor- und Nachteile

- **Symmetrische Kryptosysteme:**

  + hohe Geschwindigkeit (z.B. DES 1000 mal schneller als RSA)

  - Problem der Schlüsselvergabe

- **Asymmetrische Kryptosysteme:**

  + öffentlicher Schlüssel darf bekanntgegeben werden

  + privater Schlüssel bleibt geheim

  + digitale Unterschrift

  - hoher Rechenaufwand => niedriger Datendurchsatz

- **Lösung:** hybride Verschlüsselung

  - Über ein Public-Key-Verfahren wird sich auf ein symmetrisches Kryptosystem mit gemeinsamen Schlüssel geeinigt.
  - Die Übertragung der Daten erfolgt mit symmetrischer Verschlüsselung.

Vorraussetzungen

- **Vorraussetzungen für asymmetrische Kryptosysteme:**
  - Sei KU der öffentliche Schlüssel und KR der private Schlüssel, so gilt für jede Nachricht m:
    KU(KR(M)) = KR(KU(M)) = m
  - Der private Schlüssel KR sollte aus dem öffentlichen Schlüssel nicht zu erschließen sein

Vertraulichkeit

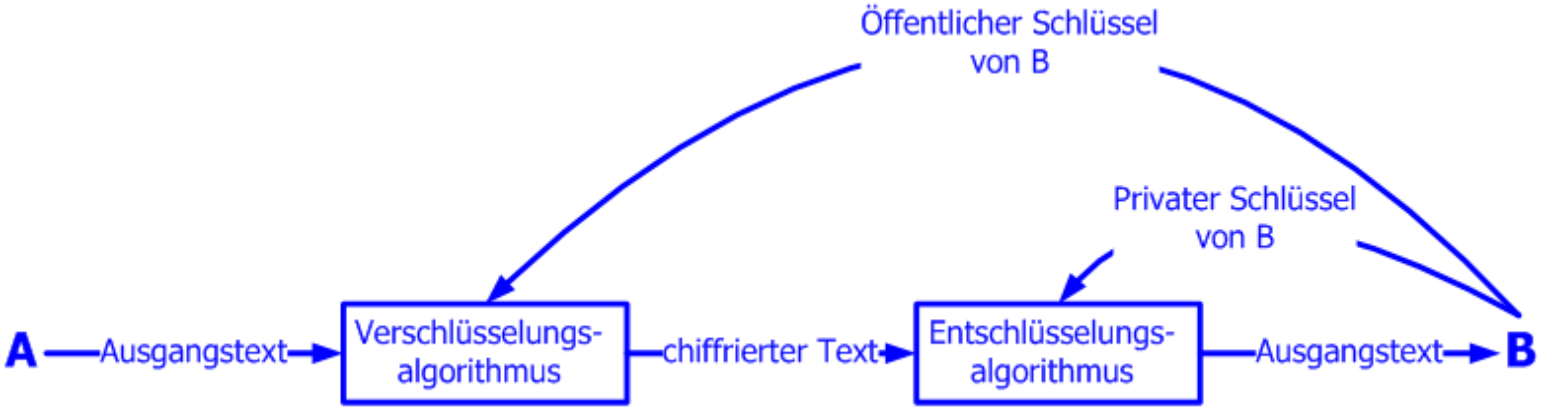


Abbildung 2: Vertraulichkeit

Authentizität

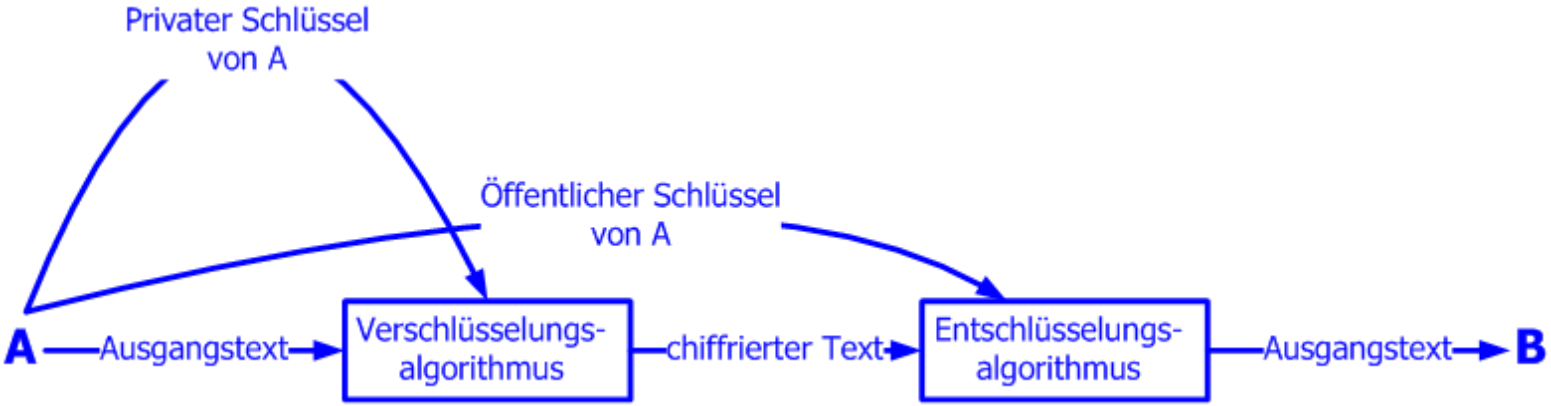


Abbildung 3: Digitale Unterschrift

Vertraulichkeit und Authentizität

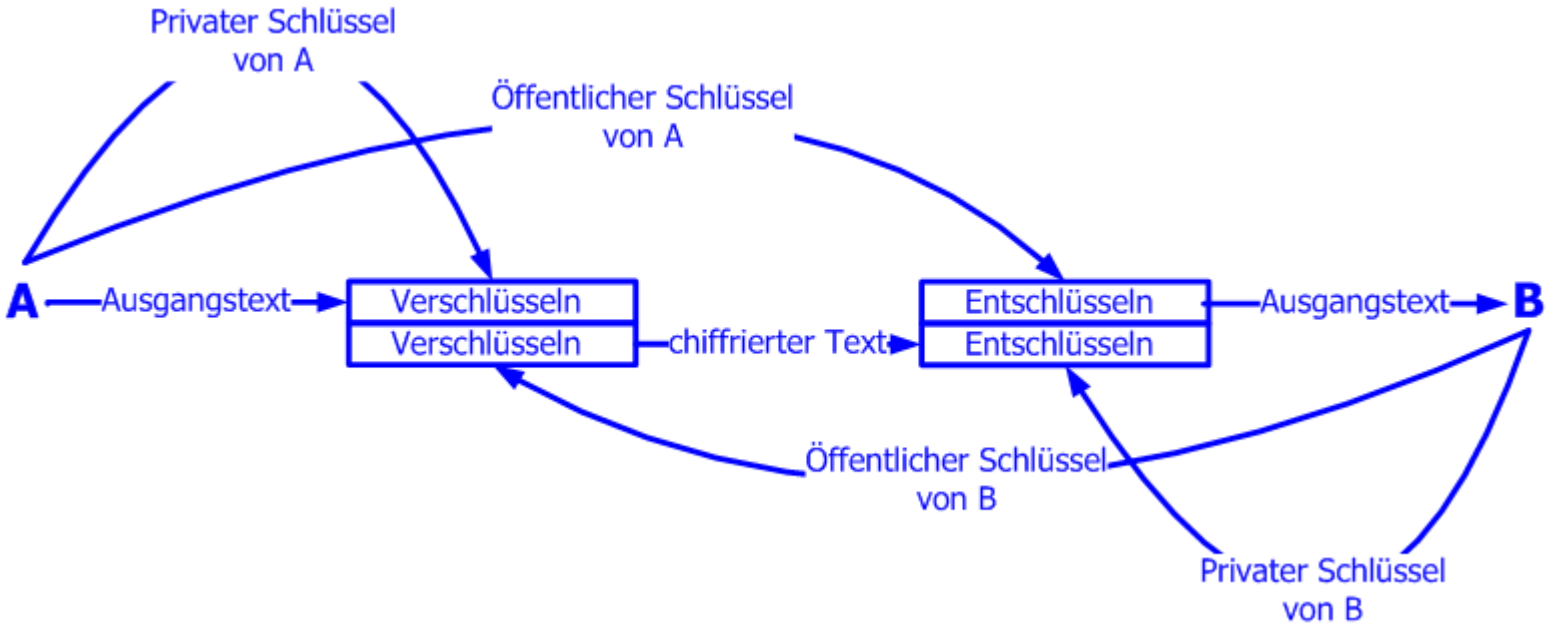


Abbildung 4: Vertraulichkeit und Authentizität

Der RSA-Algorithmus

Schlüsselgenerierung

Wähle p,q	p und q sind zwei unterschiedliche Primzahlen
Berechne $n = p \times q$	
Berechne $\phi(n) = (p-1) \times (q-1)$	
Wähle die ganze Zahl e	$ggT(\phi(n),e) = 1;\ 1 < e < \phi(n)$
Berechne d	$d = e^{-1} \bmod \phi(n)$
Öffentlicher Schlüssel	$KU = \{e,n\}$
Privater Schlüssel	$KR = \{d,n\}$

Ver- und Entschlüsselung

Ausgangstext: $M < n$

chiffrierter Text: $C = M^e \pmod n$

Chiffrierter Text: C

Ausgangstext $M = C^d \pmod n$

Der RSA-Algorithmus

Schlüsselgenerierung

Wähle p,q	p und q sind zwei unterschiedliche Primzahlen
Berechne n=p x q	
Berechne $\phi(n) = (p-1) \times (q-1)$	
Wähle die ganze Zahl e	$ggT(\phi(n),e)=1;\ 1<e<\phi(n)$
Berechne d	$d= e^{-1} \bmod \phi(n)$
Öffentlicher Schlüssel	KU = {e,n}
Privater Schlüssel	KR = {d,n}

Ver- und Entschlüsselung

Ausgangstext: $M<n$

chiffrierter Text: $C = M^e \pmod n$

Chiffrierter Text: C

Ausgangstext $M= C^d \pmod n$

Die Eulersche ϕ-Funktion (1/2)

- Zahlentheoretische Funktion

> Die Eulersche ϕ-Funktion gibt für jede natürliche Zahl n an, wieviele natürliche Zahlen zwischen 1 und n zu ihr teilerfremd sind. Zwei Zahlen a und b sind zu einander teilerfremd, gdw. ggT(a,b) = 1.

  - $\Rightarrow$ Sei n eine Primzahl, so sind alle Zahlen von 1 bis n-1 zu n teilerfremd
  - Somit ergibt sich für p und q: $\phi(p) = p - 1$ und $\phi(q) = q - 1$

- Des Weiteren gilt:

> Die Eulersche ϕ-Funktion ist multiplikativ.
> Für teilerfremde Zahlen, das heißt Zahlen m und n mit ggT(m,n) = 1, gilt damit: $\phi(mn) = \phi(m)\phi(n)$

  - $\Rightarrow \phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$

Die Eulersche ϕ-Funktion (2/2)

- **Beispiel:**
  - p = 7
  - q = 11
  - n = 77
  - $\phi(n) = (7-1)(11-1) = 60$

Der RSA-Algorithmus

Schlüsselgenerierung

Wähle p,q	p und q sind zwei unterschiedliche Primzahlen
Berechne $n = p \times q$	
Berechne $\phi(n) = (p-1) \times (q-1)$	
Wähle die ganze Zahl e	**$ggT(\phi(n),e) = 1;\ 1 < e < \phi(n)$**
Berechne d	$d = e^{-1} \bmod \phi(n)$
Öffentlicher Schlüssel	$KU = \{e,n\}$
Privater Schlüssel	$KR = \{d,n\}$

Ver- und Entschlüsselung

Ausgangstext: $M < n$

chiffrierter Text: $C = M^e \pmod n$

Chiffrierter Text: C

Ausgangstext $M = C^d \pmod n$

Der Euklidische Algorithmus (1/2)

Der Euklidische Algorithmus basiert auf folgenden Theorem:
Für jede nicht negative ganze Zahl a und eine beliebige positive Zahl b gilt: ggT(a,b) = ggT(b, a mod b)
Desweiteren gilt: ggT(a, 0) = a

- **Beispiele:**
  - ggT(12, 18) = ggT(18, 12) = ggT(12, 6) = ggT(6, 0) = 6
  - ggT(10, 11) = ggT(11, 10) = ggT(10, 1) = ggT(1, 0) = 1

Der Euklidische Algorithmus im Pseudocode:

```
geg: a, b

while(b>0){
     r = a mod b;
     a = b;
     b=r;
}
ggT = a;
```

Der Euklidische Algorithmus (2/2)

- **Beispiel für RSA:**
  - Für den öffentlichen Schlüssel wird eine zu $\phi(n)$ teilerfremde Zahl gewählt.
  - Wahl: $e = 13$, denn $1 < 13 < 60$ und $ggT(e, \phi(n)) = ggT(13, 60) = 1$

Der RSA-Algorithmus

Schlüsselgenerierung

Wähle p,q	p und q sind zwei unterschiedliche Primzahlen
Berechne $n = p \times q$	
Berechne $\phi(n) = (p-1) \times (q-1)$	
Wähle die ganze Zahl e	$ggT(\phi(n),e)=1;\ 1<e<\phi(n)$
Berechne d	**$d = e^{-1} \bmod \phi(n)$**
Öffentlicher Schlüssel	$KU = \{e,n\}$
Privater Schlüssel	$KR = \{d,n\}$

Ver- und Entschlüsselung

Ausgangstext: $M < n$

chiffrierter Text: $C = M^e \pmod n$

Chiffrierter Text: C

Ausgangstext $M = C^d \pmod n$

Der erweiterte Euklidische Algorithmus

Der erweiterte euklidische Algorithmus ist ein Algorithmus aus dem mathematischen Teilgebiet der Zahlentheorie.
Er berechnet neben dem größten gemeinsamen Teiler ggT(a,b) zweier natürlicher Zahlen a und b noch zwei ganze Zahlen s und t, die die folgende Gleichung erfüllen:

$$ggT(a,b) = s * a + t * b$$

- **RSA**
  - $d = e^{-1} \bmod \phi(n)$
  - $de = 1 \bmod \phi(n)$
  - $de + k\phi(n) = 1$
  - $ggT(e, \phi(n)) = d*e + k*\phi(n) = 1$
  - Beispiel:
    - $e = 13$, $\phi(n) = 60$, $ggT(13, 60) = d*13+k*60 = 1$
    - Mögliche Lösung: $d = 37$, $k = -8$
    - Denn $37*13 \bmod 60 = 1$

Der RSA-Algorithmus

Schlüsselgenerierung

Wähle p,q	p und q sind zwei unterschiedliche Primzahlen
Berechne $n = p \times q$	
Berechne $\phi(n) = (p-1) \times (q-1)$	
Wähle die ganze Zahl e	$ggT(\phi(n),e) = 1;\ 1 < e < \phi(n)$
Berechne d	$d = e^{-1} \bmod \phi(n)$
Öffentlicher Schlüssel	$KU = \{e,n\}$
Privater Schlüssel	$KR = \{d,n\}$

Ver- und Entschlüsselung

Ausgangstext: $M < n$

chiffrierter Text: $C = M^e \pmod n$

Chiffrierter Text: C

Ausgangstext $M = C^d \pmod n$

Ver- und Entschlüsselung (1/5)

Einwegfunktion

Eine Einwegfunktion ist eine leicht zu berechnende Funktion, die Berechnung ihrer Umkehrung soll unmöglich sein.
RSA verwendet eine Einwegfunktion mit Hintertür (engl. trap door one-way function), um die Umkehrung der Funktion durch eine Zusatzinformation und damit die Entschlüsselung zu ermöglichen.

Modulares Potenzieren

[(a mod n) * (b mod n)] mod n = (a*b) mod n

Pseudocode:

```
geg: Nachricht m, Schlüssel key und n als natürliche Zahlen
          p = 1;
            while (key > 0){
                if (key mod 2 == 1)p = p * m mod n;
                m = m * m mod n;
                key = key / 2;
            }
            return p;
        }
```

Ver- und Entschlüsselung (2/5)

- **Modulares Potenzieren**
  - Beispiel für key = 5:

    key = 5 **m** * m * m * m* m

    key = 2 m^2 * m^2

    key = 1 m^4
  - Es werden weniger als $2 \log_2(\text{key})$ Multiplikationen benötigt.

Ver- und Entschlüsselung (3/5)

- **Satz von Euler-Fermat:** $\forall a \in (\mathbb{Z}_n, *): a^{\phi(n)} \equiv 1 \, mod \, (n)$

  - Wofür? Es muß gewährleistet sein, daß das Verfahren auch umgekehrt funktioniert.
  - **Beweis:**

    $(\mathbb{Z}_n, *) = \{z_1, \ldots, z_{\phi(n)}\}$

    Da $az_i \not\equiv az_j \, mod \, (n)$ für $i \neq j$:

    Jedem Element von $(\mathbb{Z}_n, *)$ kann ein Faktor a mitgegeben werden.

    Die Mengendarstellung der Gruppe sieht dann so aus:

    $\{az_1, \ldots, az_{\phi(n)}\}$

    Somit gilt ebenso:

    $$\prod_{i=1}^{\phi(n)} z_i \equiv \prod_{i=1}^{\phi(n)} az_i \equiv a^{\phi(n)} \prod_{i=1}^{\phi(n)} z_i$$

    Somit muß gelten: $a^{(\phi(n))} \equiv 1 \bmod (n)$

    q.e.d

Ver- und Entschlüsselung (4/5)

Botschaft M soll verschlüsselt werden zu C:

C entsteht so:

$$C^d \bmod (n) = (M^e)^d \bmod (n) = M^{e*d} \bmod (n)$$

Die Rückformung erfolgt folgendermaßen:

$$M = C^d \bmod (n)$$

Setzt man diese Gleichungen zusammen, bleibt zu zeigen: $M = M^{(e*d)} \bmod (n)$

Aus $e*d \bmod (\varphi(n)) = 1$ kann man folgern:

$$e*d = k*\bmod (\varphi(n)) + 1$$

$M^{(e*d)}$ läßt sich schreiben als:

$$M^{(1+k*\phi(n))} = M*M^{(k*\phi(n))} = M*(M^{(\phi(n))})^k$$

Ver- und Entschlüsselung (5/5)

Da der Satz von Euler-Fermat lautet:

$a^{\phi(n)} = 1 \bmod(n)$, gilt somit:

$M*(M^{\phi(n)})^k = M*1^k = M$

q.e.d.

Die Sicherheit des RSA-Verfahrens

- Die Sicherheit von RSA basiert auf dem Problem, eine große ganze Zahl in ihre Primfaktoren zu zerlegen (Faktorisierungsproblem)

- Man muss einige Richtlinien beachten, um mit RSA sicher arbeiten zu können.

Faktorisierungsproblem (1/3)

- Wie schnell kann man faktorisieren?

Die schnellsten allgemeinen Verfahren Zahlkörpersieb und Elliptische-Kurven-Faktorisierung haben einen Zeitaufwand der Größenordnung:

$$e^{\sqrt[3]{\ln n * (\ln \ln n)^2}}$$

MIPS-Jahre, um den Schlüssel zu knacken	RSA Schlüssellänge
10^4	512 Bit
10^{11}	1024 Bit

Faktorisierungsproblem (2/3)

- Was kann ein PC?

Sekunden/Jahr	$30 * 10^6$
1 MIPS-Jahr	$30 * 10^{12}$ Instruktionen
386-Architektur (32 Bit)	ca. 0.5 Instruktionen/Takt
2-GHz-PC	10^9 Instruktionen/Sekunde = 10^3 MIPS = 1 GIPS
... benötigt für 1 MIPS-Jahr	$30 * 10^3$ Sekunden = 500 Minuten = 8:20 Stunden
... für 100 MIPS-Jahre (400-Bit-Modul)	≈ 35 Tage
... für 512-Bit-Modul	**400 Wochen ≈ 8 Jahre**

Faktorisierungsproblem (3/3)

- 512-Bit-RSA-Schlüssel sind nicht mehr sicher!
- Empfehlung

Bis Ende 2005	Bis Ende 2006	Bis Ende 2007
1024	1024 (Mindestwert) 2048 (Empfehlung)	1536 (Mindestwert) 2048 (Empfehlung)

Richtlinien

- p und q müssen groß genug sein
- p und q unterscheiden sich um wenige Bits in der Länge
  - Beispiel: Für ein n mit 1024 Bits sollten p und q etwa 512Bits groß sein.
- p – q darf nicht klein sein (Faktorisierungsangriff)
  - Wenn p - q klein ist , dann $p \approx q \rightarrow p \approx \sqrt{(n)}$
- Starke Primzahlen auswählen
  - p – 1 hat großen Primfaktor r $\Leftarrow$ Pollard p – 1 faktorisierung
  - p + 1 hat großen Primfaktor $\Leftarrow$ p + 1 Faktorisierungalgorithmus
  - r – 1 hat großen Primfaktor $\Leftarrow$ cycling attacks
  - Random p, q hat gute Eigenschaft im allgemeinen

Angriffe auf RSA (1/7)

- Faktorisierung Attack
- Brute Force Attack auf dem Privaten Schlüssel
- Small encryption exponent e
- Forward Search Attack
- Small decryption exponent d
- Homomorphieeigenschaft
- Common modulus Attack
- Cycling Attack
- Quantum Computer Attack

Angriffe auf RSA (2/7)

- **Faktorisierung Attack**
  - Primfaktorzerlegung einer natürlichen Zahl

Besonders schnell zum Ziel, wenn $p \approx q \rightarrow p \approx \sqrt{(n)}$

$$n = a^2 - b^2 = \underbrace{(a+b)}_{p} \cdot \underbrace{(a-b)}_{q} = p \cdot q$$

- **Brute Force Attack auf d**
  - Alle möglichen d nacheinander ausprobieren

Angriffe auf RSA (3/7)

Small encryption exponent e

A sendet einen Klartext M zu B,C,D mit dem Schlüssel (e, n_i) und $e = 3$:

$$c_i = m^3 \bmod n_i,\ i = 1,2,3$$

Eve bekommt c_1, c_2, c_3.

Gauss'scher Algorithmus: $0 \leq x \leq n_1 n_2 n_3$

$$x \equiv c_1 \pmod{n_1}$$
$$x \equiv c_2 \pmod{n_2}$$
$$x \equiv c_3 \pmod{n_3}$$

$$n_1 n_2 n_3 \rightarrow x = m^3 \rightarrow \sqrt[3]{x}$$

Lösung: Für jedes M verschiedenes Salz verwenden.

Angriffe auf RSA (4/7)

Forward Search Attack

Wenn Cyphertext C zu klein ist, kann Eve ein Wörterbuch für alle M erstellen.

Lösung: Für kleines M „Salz“ verwenden.

Small decryption exponent d

Wenn $|d| = \frac{1}{4}|n|$, kann man d leicht finden.

Lösung: d sollte fast genauso groß wie n sein.

Angriffe auf RSA (5/7)

Homomorphieeigenschaft

$sig_1 sig_2 = m_1^e m_2^e \bmod n = (m_1 m_2)^e \bmod n$

sig3 = sig1sig2 – eine neue Signatur

Lösung: Einen Hashwert benutzen.

Common modulus attack

Jede Person sollte ein unterschiedliches n benutzen, da die Verwendung des gleichen n's Risken in sich birgt.

Mit Faktorisierung von n auf p und q, kann A alle verschlüsselte Nachrichten von B,C,... knacken.

Angriffe auf RSA (6/7)

Cycling Attack

$c^{e^1} \bmod n, c^{e^2} \bmod n, \ldots, c^{e^{k-1}} \bmod n, c^{e^k} \bmod n = c \bmod n$

$$c^{e^{k-1}} \equiv m \pmod{n}$$

Generalisierte cycling attack

Finde den kleinsten positiven integer u, so dass

$f = ggT(c^{e^u} \bmod n - c, n) > 1$

$c^{e^u} \equiv c \bmod p$ and $c^{e^u} \not\equiv c \bmod q \rightarrow f = p$

$c^{e^u} \not\equiv c \bmod p$ and $c^{e^u} \equiv c \bmod q \rightarrow f = q$

$c^{e^u} \equiv c \bmod p$ and $c^{e^u} \equiv c \bmod q \rightarrow f = n$:

$$c^{e^{k-1}} \equiv m \pmod{n}$$

Basic cycling attack (weniger häufig)

Die Generalisierte Cycling attack ist genauso schwer wie die Faktorisierung von n.

Angriffe auf RSA (7/7)

- **Quantum Computer attack**
  - Ein Quantencomputer könnte das Faktorisierungsproblem lösen und sehr schnell die Werte für p und q knacken.
  - Peter Shor hat einen Algorithmus erfunden, der auf einem Quantencomputer benutzt werden kann, um RSA zu knacken.
  - Dieser Algorithmus wurde auf einem Quanten- computer ausgeführt und hat RSA für n = 15 mit p = 3 und q=5 geknackt.

Digitale Signatur (1/2)

- **Idee**

  Eine Einwegfunktion macht noch keine Digitale Signatur

  . . . sondern braucht als Ergänzung noch die asymmetrische Verschlüsselung eines Public-Key-Verfahrens.

- **Eigenschaften**
  - Eindeutigkeit (nur vom Urheber erstellbar)
  - Von jedem überprüfbar
  - Stellt die Authentizität des Unterschreibers sicher
  - Untrennbar mit dem Dokument verbunden bzw. eine Trennung von Unterschrift und das Dokument ist sofort erkennbar

Digitale Signatur (2/2)

- Absender generiert einen Hash-Wert (z.B. SHA-1) von der Datei.
- Absender verschlüsselt mit dem Geheimschlüssel (d,n) nur den Hashwert, und damit hat man die Digitale Signatur:

$$S = M^d \bmod n$$

- Empfänger dekodiert die Signatur mit dem Public key (n,e):

$$M = S^e \bmod n$$

- Der Empfänger generiert mit der gleichen Hash-Funktion einen Hashwert von dem Dokument.
- Der Empfänger vergleicht die zwei Hashwerte und prüft sie auf Gleichheit.

Anwendungen mit RSA

- Internet- u. Telefonie-Infrastruktur: X.509-Zertifikate
- Übertragungs-Protokolle: IPSec
  SSL
  TLS
  SSH
  WASTE
- E-Mail-Verschlüsselung: PGP
  S/MIME
- Authentifizierung französischer Telefonkarten

Zusammenfassung

- Asymmetrisches Verschlüsselungsverfahren
- Benutzt öffentliche und geheime Schlüssel
- Nachrichten können nur von einem Empfänger gelesen, aber vielen Absendern versandt werden
- Algorithmus beruht auf der scheinbaren Unmöglichkeit der Zerlegung von großen Produkten in Primfaktoren
- Verschlüsselung und Entschlüsselung erfolgen nach einfachen Formeln
- Am meisten benutztes Kryptographie-System
- Digitale Signaturen

10. Quellen

[1] „Network and Internetwork Security“

William Stallings

[2] „Handbook of Applied Cryptography“

Alfred J. Menezes, Paul C. van Oorschot

[3] „IT-Sicherheit“

C. Eckert

[4] „Applied Cryptography“

Bruce Schneier

[5] http://www.staff.uni-mainz.de/pommeren/DSVorlesung/KryptoBasis/RSAsich.html

[6] http://de.wikipedia.org/wiki/RSA-Kryptosystem

[7] http://www.quantencomputer.de/

[8] http://de.wikipedia.org/wiki/RSA-Kryptosystem

[9] https://www.uni-koblenz.de/~steigner/seminar-asym-krypt/distelrath-slides.pdf

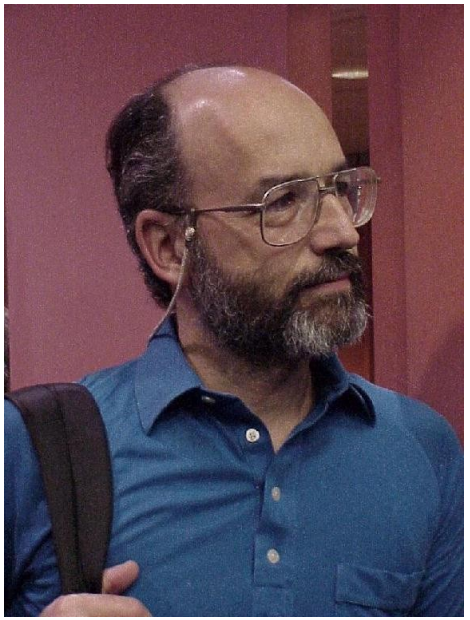
[10] http://www.cs.uni-potsdam.de/ti/lehre/05-Kryptographie/slides/RSA.pdf

[11] http://www-fs.informatik.uni-tuebingen.de/~reinhard/krypto/German/2.2.d.html

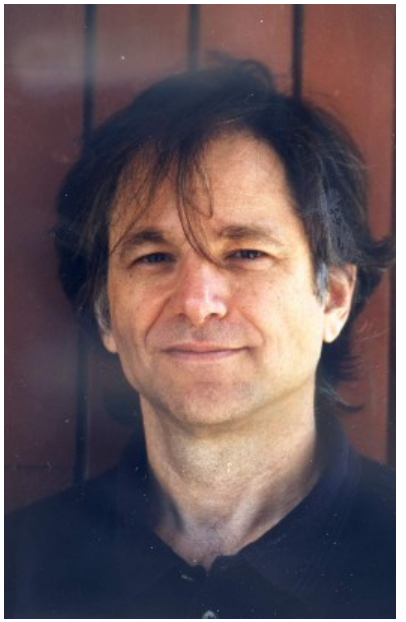
[12] http://www.inf.fh-flensburg.de/lang/krypto/algo/modexp.htm

Die Erfinder des RSA

Ron Rivest



Adi Shamir



Led Adleman