RSA Verfahren

Ghazwan Al Hayek
Hochschule für Technik Stuttgart

2. November 2008

Inhaltsverzeichnis

1 Einleitung

1.1. Übersicht

Verschlüsselung wird gebraucht, um vertrauliche Daten und Nachrichten auszutauschen oder vertrauliche Informationen sicher zu speichern.
Verschlüsselungstechniken, die Vertraulichkeit ermöglichen .sind sehr wichtig geworden währen der Entwicklung den Netzwerke allgemein und besonders das Internet.
Wenn (A) an (B) Nachrichten schicken will und will, dass keiner dieser Nachrichten lesen (verstehen) kann, muss (A) dafür sorgen, dass die Nachricht verschlüsselt wird.
Dafür benötigt (A) einen Schlüssel (e) zum Verschlüssseln die Daten.
(B) braucht den zugehörigen Schlüssel (d) zum Entschlüsseln die Daten.
Wenn der Verschlüsselungsschlüssel (e) mit dem entsprechenden Entschlüsselungsschlüssel übereinstimmt oder (d) leicht und schnell von (e) zu berechnen ist, spricht man dann von einem symmetrischen Verschlüsselungsverfahren.
Beispiel: Private-Key verfahren
Bei asymmetrischen Verschlüsselungsverfahren sind (e) und (d) im Gegensatz zu dem symmetrischen Verschlüsselungsverfahren von einander unabhängig oder genauer gesagt praktisch unmöglich (e) von (d) zu berechnen.
Beispiel: Public-Key verfahren.

1.2. Private-Key Verfahren

Unter Private-Key Verfahren versteht man dass, es nur ein Schlüssel gibt, der zum Verschlüsseln und auch Entschlüsseln gebraucht wird.
Bei Private-Key Verfahren soll die Sicherheit nicht auf der Geheimhaltung des Verfahrens beruhen, sondern nur auf der Geheimhaltung des Schlüssels.
Schlüssel können recht einfach gewechselt werden, dafür benötigt man aber ein sicherer vertraulicher Weg zum Schlüssel Austausch zwischen (A) und (B).
Bei Private-Key verfahren ergeben größere Schlüssellänge mehr Sicherheit gegen Attacke, die Probier Prinzip basieren.

1.3. Public-Key Verfahren

Bei Public-Key Verfahren muss man nur den Entschlüsselungsschlüssel geheim halten.
Er heißt geheimer Schlüssel oder private-Key und der entsprechende Verschlüsselungsschlüssel kann man veröffentlichen.
Er heißt öffentlicher Schlüssel oder public-Key.
Jeder Teilnehmer schreibt seinen öffentlichen Schlüssel in einem öffentlichen Verzeichnis.
Will einer an ihm eine verschlüsselte Nachricht schicken, so besorgt er sich diesen öffentlichen Schlüssel aus dem Verzeichnis, verschlüsselt damit die Nachricht und schickt sie.

1.4. Vor-/ Nachteile der public-/ Private key Verfahren

Symmetrische Verfahren haben den Nachteil, dass sie einen sicheren Kanal benötigen, um überhaupt eine sichere Verschlüsselung zu ermöglichen.
Es bleibt noch folgendes Problem einiger symmetrischer Verfahren besehen: die Anzahl der Schlüssel. Als Beispiel diene das Netzwerk einer Firma mit n Teilnehmer. Wollen diese in einer Sitzung paarweise geheim miteinander kommunizieren, so kann dies durch den Austausch geheimer Schlüssel zwischen jeweils zweiten der Teilnehmer geschehen. Dabei müssen je Sitzung $n(n – 1)/2$ Schlüssel geheim ausgetauscht werden, und entsprechend viele müssen geschützt gespeichert werden. Mit ansteigender Anzahl von Teilnehmer (wie im Internet) wird klar, dass dieser Schlüsselaustausch nicht leicht zu organisieren ist.
Dieses Problem wird durch den Ansatz von Public-Key-Verfahren einfacher:
Es brauchen keine Schlüssel mehr ausgetauscht zu werden.
Jedoch muss berücksichtigt werden, dass das öffentliche Schlüsselverzeichnis, in dem von jedem Teilnehmer der öffentlichen Schlüssel aufgelistet ist, von den möglichen Attacke geschützt werden muss.
Das heißt ein Angreifer kann Schlüssel ersetzen und so Nachrichten, die nicht für ihn bestimmt sind, lesen.
Da die Publik-Key-Verfahren nicht so effizient wie viele symmetrische Verfahren sind, benutzt man in der Praxis Kombination aus symmetrischen und asymmetrischen Verfahren, wie Hybrid-Verfahren.

1.5. Hybrid-Verfahren

Bei Hybrid-Verfahren wird das Public-Key-Verfahren nur zum Schlüsselaustausch verwendet. Gegeben ist ein Dokument x, das verschlüsselt werden soll. Dazu wird ein sogenannter Sitzungsschlüssel erzeugt. dieser wird nur zur Verschlüsselung von x benutzt und danach für keine weitere Verschlüsselung. Mit Hilfe der Verschlüsselung von x durch den Sitzungsschlüssel erhält man den Chiffretext y. Jetzt wird der Sitzungsschlüssel mit einem Public-Key-Verfahren verschlüsselt und in verschlüsselte Form an den Chiffretext angehängt.
Der Empfänger kennt den Sitzungsschlüssel nicht. Er entschlüsselt diesen also im ersten Schritt. Im zweiten Schritt entschlüsselt er mit dem Sitzungsschlüssel den Chiffretext y und erhält das Dokument x.

2 Relevante mathematische Grundlagen für RSA

2.1. Primzahlen

Bei RSA damit Schlüssel erzeugt werden können, werden zufällige Primzahlen mit fester Bit-länge K benötigt.
Primzahlen sind natürliche Zahlen, die größer 1 sind und nur durch 1 und sich selbst teilbar ohne Rest sind
Beispiel: P={2,3,5,7,11,13,17,19,23,29,31,37, …}
Es gibt unendlich viele Primzahlen.
Jede natürliche Zahl kann in eine eindeutige Reihenfolge von Primzahlen zerlegt werden.
Beispiel: $700 = 2 \cdot 2 \cdot 5 \cdot 5 \cdot 7$, $562309 = 11 \cdot 17 \cdot 97 \cdot 31$
Es gibt keine andere Möglichkeit mit anderen Faktoren auf dieses Produkt zu kommen, wenn alle Faktoren Primzahlen sein sollen.

2.2. Größte gemeinsamer Teiler ggT

Eine Zahl t nennt sich gemeinsamer Teiler von a , b wenn gilt :
t teilt a und t teilt b (t | a , t | b)
die Zahl nennt sich größte gemeinsamer Teiler wenn von a , b wenn gilt :
t | a und t | b und alle weiteren gemeinsame Teiler von a und b sind auch Teiler von t
Beispiel :ggT(8 , 12) → Teiler von 8 : 1 , 2 , 4 , 8
→ Teiler von 12 : 1 , 2 , 3 , 4 , 6 , 12
→ ggT(8 , 12) = 4
die Zahlen a , b ∈ Z (Ganze Zahlen) \ {0} heißen Teilerfremd wenn ggT(a , b) =1
Beispiel: ggT(4 , 9) → Teiler von 9: 1 , 3 , 9
Teiler von 4: 1 , 2 , 4
→ggT(1,4) =1

Wenn p eine Primzahl und $z = x + y$ ist und p teilt z dann p teilt x + y, dann ist p auch Teiler von x oder y. Das heißt teilt eine Primzahl ein Produkt zweier ganzer Zahlen, so ist dieser Primzahl auch Teiler von mindestens einen Faktor.

2.3. Kongruenzen:

Sei m eine ganze Zahl ungleich 0.
Erhält man bei Division der zwei Zahlen a , b durch m den gleichen Rest, nennt man a und b kongruent Modulo m
Also $a \equiv b \bmod (m)$
Beispiel: $3 \equiv 18 \bmod 5$ da 3/5 = 0 mit Rest = 3 und 18/5 = 3 mit Rest 3
Man kann für die Kongruenz $a \equiv b \bmod (m)$ auch die Gleichung $a = t \cdot m + b$
(b ist eine beliebige ganze Zahl) schreiben.

2.4. Restklasse

Mit Kongruenzen können wir jetzt Restklassen definieren:
Die Restklasse (a modulo m) besteht aus allen ganzen Zahlen, die bei Division den gleichen Rest wie a lassen.

Die Restklassen $[a]_m$ und $[b]_m$ sind gleich, wenn gilt $a \equiv b \bmod(m)$
Für RSA werden Prime Restklassen benötigt. Man spricht von einer prime Restklassen wenn eine Restklasse $[a]_m$ den ggT(a , b) = 1, das heißt a und b müssen Teilerfremd sein.

2.5. Eulersche φ-Funktion :

φ(n) : Anzahl der zu n Teilfremden a mit $1 < a < n$
Beispiel : φ(12) = 4 , φ(10) = 4 , φ(p) = p– 1 : p primzahl

2.6. Satz von Euler-Fermat

Für zwei Zahlen a , p mit der Eigenschaften :
P = primzahl und p teilt a nicht , gilt folgende Kongruenz

$$a^{p-1} \equiv 1 \bmod (p)$$

zusammen mit φ-Funktion **: $a^{\varphi(m)} \equiv 1 \bmod (m)$**

Beispiel : a = 5 , m=8 → φ(8) = 4 , ggT(5 , 8) = 1 → $5^4 \equiv 1 \bmod 8$
($5^4 = 625$ → 625 / 8 = 78 mit Rest 1)

2.7. Euklidische Algorithmus

Seien $a,b \in N$ und $a > b$. . Um ggT(a,b) zu finden, mache sukzessiv:
$a = b \cdot q_1 + r_1 \qquad 0 < r_1 < b$
$b = r_1 \cdot q_2 + r_2 \qquad 0 < r_2 < b$
$r_1 = r_2 \cdot r_q + r_3 \qquad 0 < r_3 < b$
.
.
.

Bis $r_k = 0$. dann ist r_{k-1} = ggT(a , b)
Satz: Der Euklidische Algorithmus „funktioniert“, d.h. er terminiert nach endliche vielen Schritten
(d.h. nach endlich vielen Schritten ist tatsächlich $r_k = 0$) und r_{k-1} = ggT(a , b).
Beispiel: ggT(1547 , 560)

$$\begin{aligned}
1547 &= 2 \cdot 560 + 427 \\
560 &= 1 \cdot 427 + 133 \\
427 &= 3 \cdot 133 + 28 \\
133 &= 4 \cdot 28 + 21 \\
28 &= 1 \cdot 21 + \mathbf{7} \quad \leftarrow = \text{ggT} \\
21 &= 3 \cdot 7 + 0
\end{aligned}$$

→ ggT(1547 , 560) = 7

2.8. Erweiterter Euklidischer Algorithmus

Zum Lösen diese Gleichung ax + by = n. Sei $x_0 = 1$, $x_1 = 0$, sowie $y_0 = 0$, $y_1 = 1$, und $x_{i+1} = q_i \bullet x_i + x_{i-1}$, $y_{i+1} = q_i \bullet y_i + y_{i-1}$. wobei die q_i aus dem Euklidischen Algorithmus stammen, dann gilt:

$$(-1)^k \bullet x_k \bullet a + (-1)^{k+1} \bullet y_k \bullet b = ggT(a , b)$$

Beispiel: a = 1547 , b= 560

	(xi)	(yi)	(K)
	1	0	0
1547 = 2·560 + 427	0	1	1
560 = 1·427 + 133	1	2	2
427 = 3·133 + 28	1	3	3
133 = 4·28 + 21	4	11	4
28 = 1·21 + 7	17	47	5
21 = 3·7 + 0	21	58	6

k= 6 → Nach Satz: $(-1)^6 \bullet x_6 \bullet a + (-1)^{6+1} \bullet y_6 \bullet b = ggT(a , b)$

$21 \cdot 1547 - 58 \cdot 560 = 7$

$32487 - 324800 = 7$ Stimmt

3 Algorithmus und Sicherheit von RSA

3.1. Algorithmus von RSA

- wir wählen zwei große Primzahlen p und q sodass p ungleich q ist (z.B. 100- 200 Stelle im Dezimalsystem)
  sein $n = p \bullet q$ → $\varphi(n) = (p - 1) \bullet (q - 1)$

- wähle e mit $1 < e < \varphi(n)$, und ggT $(e , \varphi(n)) = 1$, sodass
  $d \bullet e \equiv 1 \bmod \varphi(n)$ Dann :
  - (e , n) ist der Public-Key
  - d ist der Private-Key
  - n ist der RSA modul

- **Verschlüsselung**
  Sei der Klartext eine natürliche Zahl m mit $0 < m < n$. dann ist der Chiffretext: $\boldsymbol{m^e \bmod n}$

- **Entschlüsselung**
  Ist c der Chiffretext, also $c \in \mathbf{N}$ mit $0 < c < n$ dann bestimme $\boldsymbol{c^d \bmod (n)}$

3.2. RSA-Beispiel:

Wählen wir p = 13, q = 23 (unrealistisch klein)
$n = p \bullet q = 299$
$\varphi(n) = (p - 1) \bullet (q - 1) = 12 \bullet 22 = 264 = 8 \bullet 3 \bullet 11$
Kleinmöglichste e = 5 → Öffentlicher Schlüssel (n , e) = (299 , 5).

Bestimmung von d mit $e \bullet d \equiv 1 \bmod \varphi(n)$
Von dem erweiterten Euklidischen Algorithmus mit $\varphi(n) = 264$ und $e = 5$ kommen wir zu der folgenden Gleichung :$1 = ggT(264,5) = 264 \bullet (-1) + 5 \bullet 53$
$\rightarrow d = 53$ ($e \bullet d = 5 \cdot 53 \equiv 1 \bmod 264$)
Geheimer Schlüssel (n , d) = (299 , 53)
Der Klartext sei m = 212.
Zur Verschlüsselung muss man $m^5 \bmod 299$ berechnen.

$212^2 \equiv 44944 \equiv 94 \bmod 299$
$212^4 \equiv 942 \equiv 8836 \equiv 165 \bmod 299$
$212^5 \equiv 165 \cdot 212 \equiv 34980 \equiv 296 \bmod 299$

Verschlüsselter Text ist c = 296.
Zur Entschlüsselung wird $c^{53} \bmod 299$ berechnet.

$53 = 2^5 + 2^4 + 2^2 + 2^0$, daher
$296^{53} = ((((296^2 \cdot 296)^2)^2 \bullet 296)^2)^2 \cdot 296.$

$296^2 \equiv (-3)^2 \equiv 9 \bmod 299$
$296^2 \bullet 296 \equiv 9 \cdot (-3) \equiv -27 \bmod 299$
$(-27)^2 \equiv 729 \equiv 131 \bmod 299$
$(-27)^4 \equiv 131^2 \equiv 17161 \equiv 118 \bmod 299$
$(-27)^4 \bullet 296 \equiv 118 \cdot (-3) \equiv -354 \equiv 244 \bmod 299$
$244^2 \equiv (-55)^2 \equiv 3025 \equiv 35 \bmod 299$
$244^4 \equiv 35^2 \equiv 29 \bmod 299$
$29 \cdot 296 \equiv 8584 \equiv 212 \bmod 299$
Die Entschlüsselung liefert also wieder den Klartext m = 212.

3.3. Sicherheit von RSA

Die Sicherheit des RSA-Algorithmus liegt in der Schwierigkeit große Zahlen in ihre Primfaktoren zu zerlegen.
Prinzipiell Man kann das Produkt $n = p \bullet q$ sehr leicht berechnen, aber wenn man nur die Zahl n kennt, dann ist es sehr schwer aus n p und q zu ermitteln.
Dabei handelt es sich um eine Trapdoor-Einwegfunktion*. Diese Asymmetrie nutzt das RSA-Verfahren aus und darauf basiert die Sicherheit des RSA.
Wenn man in der Lage ist große Zahlen schnell zu faktorisieren, wird dann das RSA-Algorithmus nicht 100%ig sicher und so kann man es knacken.
Jedoch sind die heute verwendeten Zahlen so groß (über 200-stellig), dass es selbst mit sehr guten Faktorisierungsverfahren mit dem heutigen existierenden Computern, die sehr gute Leistung haben, nicht möglich ist, sie zu zerlegen.
Dies ist nur mit Spezialsoftware und speziell zum Faktorisieren entwickelter Hardware möglich. Das interessante an diesem Problem ist, dass es bis heute keine wirklich schnelle Methode zum Faktorisieren von großen Zahlen gibt, und da stellt sich die Frage, ob eine solche überhaupt geben kann.

***** Trapdoor-Einwegfunktion ist eine Einwegfunktion, also eine außerordentlich schwer zu invertierende Funktion, zu der es aber eine Geheiminformation gibt, mit Hilfe derer man die Funktion leicht invertieren kann.

3.4. Primfaktorzerlegung

In der Mathematik versteht man unter der Primfaktorzerlegung, auch als Zerlegung in Primfaktoren bezeichnet, die Darstellung einer natürlichen Zahl als Produkt von Primzahlen.
$6936 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 17 \cdot 17$
Die Primzahl, die in der Primfaktorzerlegung einer Zahl vorkommt nennt man die Primfaktoren dieser Zahl. Zum Beispiel hat 6936 die Primfaktoren 2, 3 und 17. Den Exponenten des jeweiligen Primfaktors p nennt man die p-Bewertung oder den p-Exponenten der Zahl.
Die Zahl 6936 hat beispielsweise den 2-Exponenten 3, den 3-Exponenten 1 und den 17-Exponenten 2. Alle anderen p-Exponenten sind gleich 0.
Effiziente Algorithmen, die die Primfaktorzerlegung einer natürlichen Zahl berechnen, sind nicht bekannt.
Es ist aber auch kein Beweis bekannt, der zeigt, dass das Faktorisierungsproblem schwer ist. Es ist daher möglich, dass es effiziente Faktorisierungsverfahren gibt ,und dass die auch schon bald gefunden werden. Dann ist das RSA-Verfahren unsicher und muss durch andere Verfahren ersetzt werden.
Der französische Jurist Pirre der Fermat (1601 bis 1665) glaubte, dass die nach ihm benannten *Fermat-Zahlen*

$$\mathbf{F_i = 2^{2i} + 1}$$

Sämtliche Primzahlen seien. Tatsächlich sind

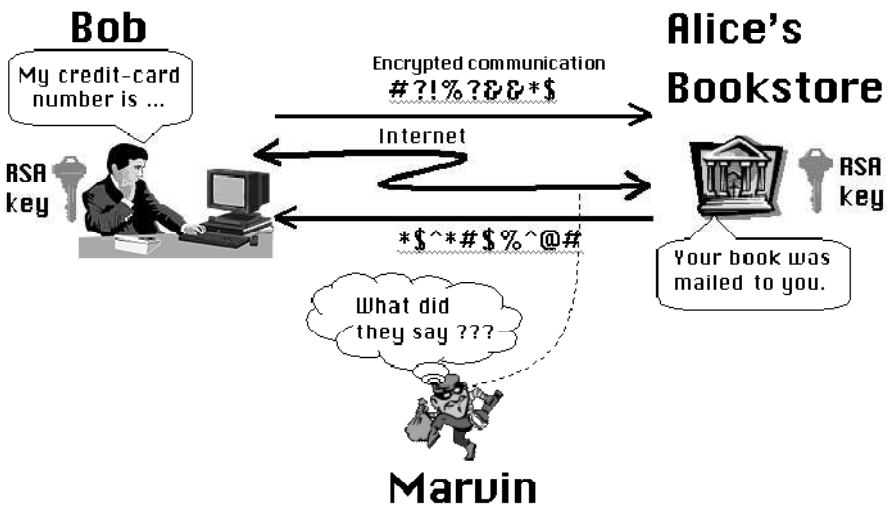
- $F_0 = 2^0 + 1 = 3$ Primzahl
- $F_1 = 2^2 + 1 = 5$ Primzahl
- $F_2 = 2^4 + 1 = 17$ Primzahl
- $F_3 = 2^8 + 1 = 257$ Primzahl
- $F_4 = 2^{16} + 1 = 65537$ Primzahl

Aber 1732 fand Euler heraus, dass $F_5 = 641 \cdot 6700417$ zusammengesetzt ist.
Die angegebene Faktorisierung ist auch die Primfaktorzerlegung der fünften Fermat-Zahl.
Auch F_6, F_7, F_8 und F_9 sind zusammengesetz .

- Die Faktorisierung von F_6 wurde 1880 von Landry und Le Lasseur gefunden
- Die Faktorisierung von F_7 wurde 1970 von Brillhart und Morrison gefunden
- Die Faktorisierung von F_8 wurde 1980 von Brent und Pollard gefunden
- Die Faktorisierung von F_9 wurde 1990 von Lenstra , Manasse und Pollard gefunden

Einerseits sieht man an diesem Daten, wie schwierig das Faktorisierungsproblem ist. Immerhin hat es bis 1970 gedauert, bis die 39-stellige Fermat-Zahl F_7 zerlegt war.
Andererseits ist die enorme Weiterentwicklung daran zu erkennen, dass nur 20 Jahre später die 155-stellige Fermat-Zahl F_9 faktorisiert wurde.

4 Mögliche Angriffe und RSA und Gegenmaßnahmen



Um Angriffe auf Verschlüsselungsverfahren zu erschweren, kann man das verwendete Verfahren geheim halten. Die Sicherheit, die man daraus gewinnen kann, ist aber sehr zweifelhaft. Ein Angreifer hat viele Möglichkeiten, zu erfahren welches Verschlüsselungsverfahren verwendet wird. Er kann verschlüsselte Nachrichten abfangen und daraus Rückschlüsse ziehen. Daher werden in öffentlichen Anwendungen, wie zum Beispiel im Internet öffentlich bekannte Verschlüsselungsverfahren benutzt. Das Militär und die Geheimdienste verwenden aber oft geheime Verschlüsselungsverfahren.

4.1. Known-plaintext-Attacke

Der Angreifer kennt eine begrenzte Anzahl von Geheimtexten mit den zugehörigen Klartexten und möchte daraus den verwendeten Schlüssel bzw. den Klartext zu einem weiteren Chiffretext berechnen.
Um Known-plaintext-Attacke zu vermeiden muss die Verschlüsselung mit Hilfe des Startwertes randomisiert.
Diese Maßnahme nennt sich Randomisierung.

4.2. Chosen-plaintext-Attacke

Da bei RSA der Verschlüsselungsschlüssel öffentlich ist, kann der Angreifer Chiffretexte zu selbst gewählte Klartexten erzeugen.
Der Angreifer fängt einen Schlüsseltext ab. Er weiß, dass der zugehörige Klartext entweder „ja“ oder „nein“ ist. Um den richtigen Klartext herauszufinden, verschlüsselt er „ja“ und „nein“ und vergleicht.
Dieses Problem wird auch durch Randomisierung gelöst.

4.3. Low-Exponent-Attacke

Die Low-Exponent-Attacke kann man immer anwenden, wenn ein Klartext m mit e zueinander paarweise teilerfremden Modulen, aber immer mit den selben Verschlüsselungsexponenten e verschlüsselt wird.

Beispiel: Eine Bank sendet an e verschiedene Kunden dieselbe verschlüsselte Nachricht. Dabei werden die verschiedenen öffentlichen Schlüssel n_i, $1 \leq i \leq e$, der Kunden benutzt, aber immer derselbe Verschlüsselungsexponent e.
Dann kann der Angreifer den Klartext m rechnen, indem er aus c die e-te Wurzel zieht. Dies ist in Polynomzeit möglich.
Man kann die Low-Exponent-Attacke verhindern, indem man die Klartextblöcke kürzer wählt, als das eigentlich nötig ist, und die letzten Bits zufällig wählt.
Dass ist es praktisch ausgeschlossen, dass zweimal der selbe Block verschlüsselt wird.
Eine andere Möglichkeit, die Low-Exponent-Attacke zu verhindern, besteht darin, eine größere Verschlüsselungsexponenten zu wählen, die aber immer noch effiziente Verschlüsselung zulassen.

4.4. Brute-Force-Attacke

Bei einer Brute-Force-Attacke wird durch Ausprobieren aller Schlüssel, aus dem jeweiligen Schlüsselraum (Der Schlüsselraum ist die Menge aller möglichen Schlüssel.), versucht den Schlüssel zu erraten. Dabei werden alle möglichen Folgen von Buchstaben, Zahlen und Sonderzeichen getestet, bis der korrekte Schlüssel gefunden wurde

4.5. Man-In-The-Middle-Attacke

Ein Angreifer sitzt physikalisch oder logisch zwischen Sender und Empfänger und ist in der Lage Nachrichten abzuhören und zu verändern, ohne, dass einer der beiden Kommunikationspartner dies bemerkt. Bei Public-Key-Verfahren, kann solch ein Angriff durch Zertifikate für Public-Keys verhindert werden. Sonst wäre folgender Angriff möglich: Wenn z.B. A an B ihren öffentlichen Schlüssel EA schickt, ersetzt I diesen durch seinen eigenen Schlüssel EI und speichert EA. Wenn nun B an A eine Nachricht schickt, verschlüsselt er diese mit EI. I hört die Nachricht ab, entschlüsselt sie, verschlüsselt sie mit EA und sendet das Ergebnis an A. So kann I die Nachricht mithören und verändern, ohne dass A und B etwas bemerken.

4.6. Replay-Attacke

Bei einer Replay-Attacke wird eine Nachricht von einem Angreifer aufgezeichnet und wiederholt gesendet. So könnte zum Beispiel eine Authentifizierung mehrfach wiederholt werden. Dabei muss der Angreifer die Nachricht nicht entschlüsseln können. Er versendet einfach die abgefangene Nachricht erneut. Zur Vermeidung dieses Angriffs kann ein Zeitstempel verwendet werden
das einfachste Beispiel für eine Replay-Attacke ist, dass ein Angreifer **A** bei einer fremden Bank **X** einen Betrag von 100 €, auf sein eigenes Konto einzahlt, das bei einer anderen Bank **Y** geführt wird.
Würde die kryptographische Nachricht keine Zeitstempel oder sonstige Variablen (zum Beispiel Zufallzahlen) enthalten, sondern hatte nur die Form

$$K_{XY}(X, Y, A, € 100),$$

So könnte A die Nachricht aufzeichnen und mehrmals an seine Bank schicken.
Diese würde seinem Konto jedes Mal 100 € gutschreiben.

Fazit

Zum Knacken von RSA ist der private Schlüssel zu berechnen.
Dies ist genau so schwierig und aufwendig wie das allgemeine Faktorisierungsproblem
2005 wurde eine 640 Bit RSA Zahl mit 193 Stellen faktorisiert. Dafür brauchten aber 302,2 Ghz Rechner über 5 Monaten Zeit
Das heißt kann man momentan sagen, dass das RSA-Verfahren immer noch sicher ist. Dies gilt wenn das Produkt aus zwei Primzahlen p und q mehr als 1024 Bit hat, und solange noch keine völlig neue Hardwaretechnik existiert, die das Faktorisierungsproblem leicht lösen kann.

Literatur

[1] Johannes Buchmann
Einführung in die Kryptographie
ISBN 3-540-40508-9

[2] Albrecht Beutelspacher, Jörg Schwenk und Klaus-Dieter Wolfenstetter
Moderne Verfahren der Kryptographie
ISBN 3-8348-0083-x

[3]Wikipedia (Autor Unbekannt), RSA-Kryptosystem
http://de.wikipedia.org/wiki/RSA-Kryptosystem
Zugrif : Oktober 2008

[4] Die Sicherheit des RSA-Verfahrens
http://www.hh.schule.de/julius-leber-schule/melatob/sicherheit.html
Zugrif : Oktober 2008