

RSA Verfahren

Ghazwan Al Hayek
Hochschule für Technik Stuttgart

15. Januar 2009

Inhaltsverzeichnis

1. Einleitung
 - 1.1. Übersicht
 - 1.2. Private-Key-Verfahren
 - 1.3. Public-Key-Verfahren
 - 1.4. Vor/ Nachteile von Public/Private-Key-Verfahren
 - 1.5. Hybrid-Verfahren
2. Relevante mathematische Grundlagen für RSA
 - 2.1. Primzahlen
 - 2.2. Größte gemeinsamer Teiler
 - 2.3. Kongruenzen
 - 2.4. Restklassen
 - 2.5. Eulersche φ -Funktion
 - 2.6. Satz von Euler-Fermat
 - 2.7. Euklidischer Algorithmus
 - 2.8. Erweiterter Euklidischer Algorithmus
3. Algorithmus und Sicherheit von RSA
 - 3.1. Algorithmus von RSA
 - 3.2. RSA-Beispiel
 - 3.3. Sicherheit des RSA
 - 3.4. Primfaktorzerlegung
4. Mögliche Angriffe und Gegenmaßnahmen
 - 4.1. Known-plaintext-Attacke
 - 4.2. Chosen-plaintext-Attacke
 - 4.3. Low-Exponent-Attacke
 - 4.4. Brute-Force-Attacke
 - 4.5. Man-In-The-Middle-Attacke
 - 4.6. Replay-Attacke
5. Fazit
- Literatur

1 Einleitung

1.1. Übersicht

Verschlüsselung wird verwendet, um vertrauliche Daten und Nachrichten auszutauschen oder um vertrauliche Informationen sicher zu speichern.

Verschlüsselungstechniken, die Vertraulichkeit ermöglichen, sind während der Entwicklung von Netzwerken und besonders des Internets sehr wichtig geworden. Will (A) eine Nachricht an (B) senden und beabsichtigt, dass diese von keimen anderen als (B) verstanden wird, so muss (A) für die Verschlüsselung dieser Nachricht sorgen. Dazu benötigt (A) einen Schlüssel (e) der die Daten verschlüsselt. (B) hingegen benötigt den zugehörigen Schlüssel (d) mit welchem er die von (A) verschlüsselten Daten entsprechend entschlüsseln kann.

Dabei gibt es zwei unterschiedliche Verschlüsselungsverfahren. Zum einen das symmetrische Verschlüsselungsverfahren. Bei diesem stimmen Schlüssel (e) und der zugehörige Entschlüsselungsschlüssel überein oder (d) kann schnell durch (e) berechnet werden. Beispiel dafür ist das „Private-Key-Verfahren“.

Bei dem anderen Verfahren handelt es sich um das asymmetrische Verschlüsselungsverfahren. Hier sind (e) und (d), ganz im Gegensatz zum symmetrischen Verschlüsselungsverfahren, unabhängig von einander. Genauer gesagt ist es praktisch unmöglich (e) über (d) zu berechnen. Beispiel dafür ist das „Public -Key-Verfahren“.

1.2. Private-Key Verfahren

Unter Private-Key Verfahren versteht man, dass es nur ein Schlüssel gibt der sowohl zum Verschlüsseln als auch zum Entschlüsseln verwendet wird. Dieser Schlüssel, der auch geheimer Schlüssel, bzw. private-key genannt wird, kann veröffentlicht werden.

Beim Private-Key Verfahren soll die Sicherheit nicht auf der Geheimhaltung des Verfahrens, sondern nur auf der Geheimhaltung des Schlüssels beruhen.

Schlüssel können recht einfach ausgewechselt werden, man benötigt dafür aber einen sicheren und vertraulichen Weg um den Schlüssel zwischen (A) und (B) auszutauschen.

Bei diesem Verfahren gilt, dass mit der Schlüssellänge die Sicherheit gegen Attacken, die auf dem Probier-Prinzip basieren, steigt.

1.3. Public-Key Verfahren

Beim Public-Key-Verfahren muss nur der Entschlüsselungsschlüssel geheim gehalten werden.

Dieser Schlüssel wird öffentlicher Schlüssel oder auch public-Key genannt.

Dabei trägt jeder Teilnehmer seinen öffentlichen Schlüssel in ein öffentliches Verzeichnis ein.

Möchte nun jemand eine verschlüsselte Nachricht an ihn schicken, so besorgt er sich dessen öffentlichen Schlüssel aus dem Verzeichnis, verschlüsselt damit die Nachricht und sendet sie.

1.4. Vor-/ Nachteile des public-/ Private key Verfahrenes

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 bestehen: die Anzahl der Schlüssel. Als Beispiel diene das Netzwerk einer Firma mit n Teilnehmer. Wollen diese n Teilnehmer in einer Sitzung paarweise vertraulich miteinander kommunizieren, so kann dies durch den Austausch geheimer Schlüssel zwischen jeweils zwei Teilnehmern geschehen. Dabei müssen je Sitzung $n(n - 1)/2$ Schlüssel geheim ausgetauscht werden, und entsprechend viele müssen geschützt gespeichert werden. Mit ansteigender Anzahl an Teilnehmern (wie im Internet) wird klar, dass dieser Schlüsselaustausch nicht leicht zu organisieren ist.

Dieses Problem wird durch den Ansatz des Public-Key-Verfahrens vereinfacht:

Es müssen keine Schlüssel mehr ausgetauscht zu werden.

Jedoch berücksichtigt werden muss, dass das öffentliche Schlüsselverzeichnis, in dem der öffentliche Schlüssel eines jeden Teilnehmers aufgelistet ist, vor möglichen Attacken (Beispielsweise Chosen-plaintext-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 zum Beispiel Hybrid-Verfahren.

1.5. Hybrid-Verfahren

Bei Hybrid-Verfahren wird das Public-Key-Verfahren lediglich 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 anschließend für keine weitere Verschlüsselung. Mit Hilfe der Verschlüsselung von x durch den Sitzungsschlüssel, erhält man den Chiffretext y . Anschließend wird der Sitzungsschlüssel mit einem Public-Key-Verfahren verschlüsselt und in verschlüsselte Form an den Chiffretext angehängt.

Da der Empfänger diesen Sitzungsschlüssel nicht kennt, entschlüsselt er diesen also im ersten Schritt. Im zweiten Schritt entschlüsselt er mit dem zuvor entschlüsselten Sitzungsschlüssel den Chiffretext y und erhält das Dokument x .

2 Relevante mathematische Grundlagen für RSA

2.1. Primzahlen

Um RSA Schlüssel erzeugen zu können, werden zufällige Primzahlen mit fester Bit-länge K benötigt.

Primzahlen sind natürliche Zahlen, die größer 1 und ohne Rest nur durch 1 und sich selbst teilbar sind

Beispiel: $P = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, \dots\}$

Es gibt unendlich viele Primzahlen.

Jede natürliche Zahl kann in eine eindeutige Reihenfolge von Primzahlen zerlegt werden.

Beispiel: $700 = 2 \cdot 2 \cdot 5 \cdot 5 \cdot 7$, $562309 = 11 \cdot 17 \cdot 97 \cdot 31$

Da alle Faktoren aus Primzahlen bestehen sollen, gibt es keine andere Möglichkeit dieses Produkt zu erzeugen.

2.2. Größter gemeinsamer Teiler ggT

Eine Zahl t nennt sich gemeinsamer Teiler von a , b wenn gilt :

t teilt a und t teilt b ($t \mid a$, $t \mid b$)

Größter gemeinsamer Teiler von a , b nennt sich eine Zahl wenn gilt:

$t \mid a$ und $t \mid b$ und alle weiteren gemeinsame Teiler von a und b auch Teiler von t sind.

Beispiel : $\text{ggT}(8, 12) \rightarrow$ Teiler von 8 : 1 , 2 , 4 , 8
 $\rightarrow$ Teiler von 12 : 1 , 2 , 3 , 4 , 6 , 12

$\rightarrow \text{ggT}(8, 12) = 4$

die Zahlen $a, b \in \mathbb{Z} \setminus \{0\}$ heißen Teilerfremd wenn $\text{ggT}(a, b) = 1$

Beispiel: $\text{ggT}(4, 9) \rightarrow$ Teiler von 9: 1 , 3 , 9
Teiler von 4: 1 , 2 , 4

$\rightarrow \text{ggT}(4, 9) = 1$

Wenn p eine Primzahl ist, $z = x + y$ gilt und $p \mid z$ teilt, dann teilt p auch $x + y$, was heißt das p auch Teiler von x oder y ist. Das heißt, teilt eine Primzahl ein Produkt zweier ganzer Zahlen, so ist dieser Primzahl auch Teiler von mindestens einem dieser Faktoren.

2.3. Kongruenzen:

Es 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 $a \equiv b \pmod{m}$

Beispiel: $3 \equiv 18 \pmod{5}$ da $3/5 = 0$ mit Rest = 3 und $18/5 = 3$ mit Rest 3

Man kann für die Kongruenz $a \equiv b \pmod{m}$ auch die Gleichung $a = t \cdot m + b$ (b ist eine beliebige ganze Zahl) schreiben.

2.4. Restklassen

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 erzeugen.

Die Restklassen $[a]_m$ und $[b]_m$ sind gleich wenn gilt $a \equiv b \pmod{m}$

Für RSA werden Prime Restklassen benötigt. Man spricht von einer prime Restklassen wenn eine Restklasse $[a]_m$ den $\text{ggT}(a, b) = 1$ hat, was wiederum bedeutet das a und b Teilerfremd sein müssen.

2.5. Eulersche ϕ -Funktion :

$\phi(n)$: Anzahl der zu n Teilfremden a mit $1 < a < n$

Beispiel : $\phi(12) = 4$, $\phi(10) = 4$, $\phi(p) = p - 1$: 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

$$a^{p-1} \equiv 1 \pmod{p}$$

zusammen mit ϕ -Funktion : $a^{\phi(m)} \equiv 1 \pmod{m}$

Beispiel : a = 5 , m=8 $\rightarrow \phi(8) = 4$, $\text{ggT}(5, 8) = 1 \rightarrow 5^4 \equiv 1 \pmod{8}$
($5^4 = 625 \rightarrow 625 / 8 = 78$ mit Rest 1)

2.7. Euklidische Algorithmus

Seien $a, b \in \mathbb{N}$ und $a > b$. Um $\text{ggT}(a, b)$ zu finden, mache sukzessiv:

$$a = b \cdot q_1 + r_1 \quad 0 < r_1 < b$$

$$b = r_1 \cdot q_2 + r_2 \quad 0 < r_2 < r_1$$

$$r_1 = r_2 \cdot q_3 + r_3 \quad 0 < r_3 < r_2$$

.

.

.

Bis $r_k = 0$. Dann ist $r_{k-1} = \text{ggT}(a, b)$

Satz: Der Euklidische Algorithmus „funktioniert“, d.h. er terminiert nach endliche vielen Schritten

(d.h. nach endlich vielen Schritten tritt $r_k = 0$ ein) und $r_{k-1} = \text{ggT}(a, b)$.

Beispiel: $\text{ggT}(1547, 560)$

$$1547 = 2 \cdot 560 + 427$$

$$560 = 1 \cdot 427 + 133$$

$$427 = 3 \cdot 133 + 28$$

$$133 = 4 \cdot 28 + 21$$

$$28 = 1 \cdot 21 + 7 \quad \leftarrow = \text{ggT}$$

$$21 = 3 \cdot 7 + 0$$

$\rightarrow \text{ggT}(1547, 560) = 7$

2.8. Erweiterter Euklidischer Algorithmus

Zum Lösen der Gleichung $ax + by = n$ sei $x_0 = 1$, $x_1 = 0$, sowie $y_0 = 0$, $y_1 = 1$, und $x_{i+1} = q_i \cdot x_i + x_{i-1}$, $y_{i+1} = q_i \cdot y_i + y_{i-1}$, wobei q_i aus dem Euklidischen Algorithmus stammt, dann gilt:

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

Beispiel: $a = 1547$, $b = 560$	(xi)	(yi)	(K)
	1	0	0
$1547 = 2 \cdot 560 + 427$	0	1	1
$560 = 1 \cdot 427 + 133$	1	2	2
$427 = 3 \cdot 133 + 28$	1	3	3
$133 = 4 \cdot 28 + 21$	4	11	4
$28 = 1 \cdot 21 + 7$	17	47	5
$21 = 3 \cdot 7 + 0$	21	58	6

$k = 6 \rightarrow$ Nach Satz: $(-1)^6 \cdot x_6 \cdot a + (-1)^{6+1} \cdot y_6 \cdot b = \text{ggT}(a, b)$
 $21 \cdot 1547 - 58 \cdot 560 = 7$
 $32487 - 324800 = 7$ 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 $n = p \cdot q \rightarrow \varphi(n) = (p - 1) \cdot (q - 1)$
- gewählt wird e mit $1 < e < \varphi(n)$, und $\text{ggT}(e, \varphi(n)) = 1$, sodass $d \cdot e \equiv 1 \pmod{\varphi(n)}$ 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: $m^e \pmod n$
- **Entschlüsselung**
 Ist c der Chiffretext, also $c \in \mathbb{N}$ mit $0 < c < n$, dann bestimme $c^d \pmod (n)$

3.2. RSA-Beispiel:

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

Bestimmung von d mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Durch den erweiterten Euklidischen Algorithmus mit $\varphi(n) = 264$ und $e = 5$ kommen wir zu folgender Gleichung : $1 = \text{ggT}(264, 5) = 264 \cdot (-1) + 5 \cdot 53$

→ $d = 53$ ($e \cdot d = 5 \cdot 53 \equiv 1 \pmod{264}$)

Geheimer Schlüssel $(n, d) = (299, 53)$

Der Klartext sei $m = 212$.

Zur Verschlüsselung muss man $m^5 \pmod{299}$ berechnen.

$$212^2 \equiv 44944 \equiv 94 \pmod{299}$$

$$212^4 \equiv 94^2 \equiv 8836 \equiv 165 \pmod{299}$$

$$212^5 \equiv 165 \cdot 212 \equiv 34980 \equiv 296 \pmod{299}$$

Verschlüsselter Text ist $c = 296$.

Zur Entschlüsselung wird $c^{53} \pmod{299}$ berechnet.

$$53 = 2^5 + 2^4 + 2^2 + 2^0, \text{ daher}$$

$$296^{53} = (((296^2 \cdot 296)^2 \cdot 296)^2 \cdot 296).$$

$$296^2 \equiv (-3)^2 \equiv 9 \pmod{299}$$

$$296^2 \cdot 296 \equiv 9 \cdot (-3) \equiv -27 \pmod{299}$$

$$(-27)^2 \equiv 729 \equiv 131 \pmod{299}$$

$$(-27)^4 \equiv 131^2 \equiv 17161 \equiv 118 \pmod{299}$$

$$(-27)^4 \cdot 296 \equiv 118 \cdot (-3) \equiv -354 \equiv 244 \pmod{299}$$

$$244^2 \equiv (-55)^2 \equiv 3025 \equiv 35 \pmod{299}$$

$$244^4 \equiv 35^2 \equiv 29 \pmod{299}$$

$$29 \cdot 296 \equiv 8584 \equiv 212 \pmod{299}$$

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 kann man das Produkt $n = p \cdot q$ sehr leicht berechnen, kennt man aber nur die Zahl n , so ist es sehr schwierig aus n p und q zu ermitteln.

Dabei handelt es sich um eine Trapdoor-Einwegfunktion*. Eben diese Asymmetrie nutzt das RSA-Verfahren und hierauf basiert auch die Sicherheit des RSA.

Wenn man in der Lage ist große Zahlen schnell zu faktorisieren, dann wird der RSA-Algorithmus nicht 100%ig sicher und man kann ihn knacken.

Jedoch sind die heute verwendeten Zahlen so groß (über 200-stellig), dass es selbst mit den heute existierenden Computern, welche eine sehr hohe Leistungsfähigkeit aufweisen, sowie den sehr guten Faktorisierungsverfahren nicht möglich ist, diese Zahlen zu zerlegen.

Möglich ist dies nur mit einer Spezialsoftware sowie mit speziell zum Faktorisieren entwickelter Hardware. Das interessante an diesem Problem ist, dass es bis heute keine wirklich schnelle Methode zum Faktorisieren großer Zahlen entwickelt wurde; von daher stellt sich die Frage, ob es ein solches 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 Primzahlen, 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 diese auch schon bald gefunden werden. Sobald dies eintritt, wird auch das RSA-Verfahren unsicher werden und muss durch andere Verfahren ersetzt werden.

Der französische Jurist Pierre de Fermat (1601 bis 1665) glaubte, dass die nach ihm benannten *Fermat-Zahlen*

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

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

1732 fand Euler allerdings heraus, dass für $F_5 = 641 \cdot 6700417$ gilt.

Die angegebene Faktorisierung ist auch die Primfaktorzerlegung der fünften Fermat-Zahl.

Auch F_6 , F_7 , F_8 und F_9 sind zusammengesetzt .

- 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

An diesen Daten sieht man einerseits wie schwierig das Faktorisierungsproblem ist – die Zerlegung der 39-stelligen Fermat-Zahl F_7 hat immerhin bis 1970 gedauert, andererseits ist auch eine enorme Weiterentwicklung darin zu erkennen, dass nur 20 Jahre später die 155-stellige Fermat-Zahl F_9 faktorisiert wurde.

4 Mögliche Angriffe und RSA und Gegenmaßnahmen

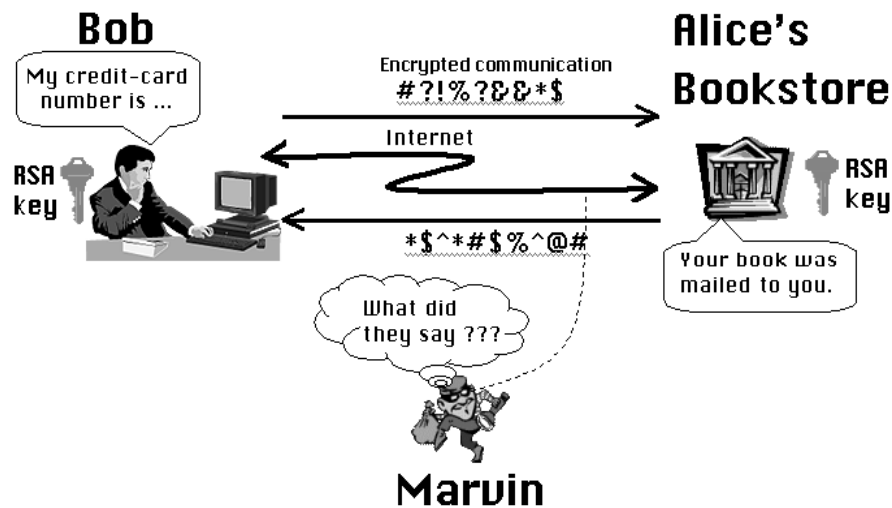


Abb. 1 Man-In-The-Middle-Angriff [2]

Um Angriffe auf Verschlüsselungsverfahren zu erschweren, kann das verwendete Verfahren geheim gehalten werden. Die daraus gewonnene Sicherheit ist aber sehr zweifelhaft, schließlich hat ein Angreifer viele Möglichkeiten um zu erfahren welches Verschlüsselungsverfahren verwendet wurde. Zum Beispiel kann er verschlüsselte Nachrichten abfangen und aus diesen Rückschlüsse ziehen. Daher werden in öffentlichen Anwendungen, wie zum Beispiel dem Internet, öffentlich bekannte Verschlüsselungsverfahren benutzt. Militär sowie Geheimdienste hingegen verwenden oft geheime Verschlüsselungsverfahren. Im Folgenden werde ich einen Einblick in 6 der am häufigsten auftretenden Attacken geben.

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.

Vom Angreifer wird ein Schlüsseltext abgefangen. Er weiß, dass der zugehörige Klartext entweder „ja“ oder „nein“ ist. Um den richtigen Klartext herauszufinden, verschlüsselt er „ja“ und „nein“ und vergleicht diese.

Dieses Problem wird auch durch Randomisierung gelöst.

4.3. Low-Exponent-Attacke

Die Low-Exponent-Attacke kann man immer dann 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 berechnen, indem er aus c die e -te Wurzel zieht. Dies ist in Polynomzeit möglich.

Um die Low-Exponent-Attacke zu verhindern ist eigentlich es eigentlich nur nötig, die Klartextblöcke kürzer und die letzten Bits zufällig zu wählen.

Es ist praktisch ausgeschlossen, dass zweimal derselbe Block verschlüsselt wird.

Eine andere Möglichkeit, die Low-Exponent-Attacke zu verhindern, besteht darin, eine vergrößerte Verschlüsselungsexponenten zu wählen, die aber eine noch immer 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 definiert als 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, wobei der Angreifer die Nachricht nicht entschlüsseln können muss. 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.

Momentan kann also gesagt werden, 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 bis eine völlig neue Hardwaretechnik existiert, die das Faktorisierungsproblem leicht lösen kann.

Literatur

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

- [2] Dan Boneh , Twenty Years of Attacks on the RSA Cryptosystem

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

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

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