

Name : Namik Ley 496200542	Sicherheit	09.07.1999
		Prof. Dr. T. Risse

1. Einleitung

Das RSA-Verfahren wurde 1978 von R. Rivest, A. Shamir und L. Adleman entwickelt. Dieser Verschlüsselungs-Algorithmus zeigte, daß Kryptographie mit public-key möglich ist. Damit gehört dieser Algorithmus zu den asymmetrischen Verfahren. D.h. zum Ver –und Entschlüsseln werden verschiedene Schlüssel verwendet. Außerdem eignet sich dieses Verfahren für digitale Signaturen, was heutzutage immer wichtiger wird.

2. Der mathematische Hintergrund des RSA-Algorithmus

Die $\phi(n)$ -Funktion (Eulerische-Funktion) gibt an, wie viele kleinere Zahlen zu n existieren, die zudem noch teilerfremd zu n sind.

Anhand der $\phi(n)$ -Funktion

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

Außerdem gilt :

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

Für $n = p \cdot q$ (p und q sind zwei verschiedene Primzahlen) gilt für dann für jede natürliche Zahl $m \leq n$ und jede natürliche Zahl k die Gleichung :

$$m^{k*(p-1)(q-1)+1} \bmod n = m$$

Bei der Realisierung des RSA-Algorithmus werden die beiden natürlichen Zahlen e und d (e =encryption-key oder auch der public-key und d =decryption-key=private-key) ermittelt. Die zuvor generierten Zahlen p und q werden nicht mehr benötigt und könnten somit vernichtet werden.

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

Aus dieser Gleichung bildet man den Modulus von $\phi(n)$

$$(e * d) \bmod \phi(n) = 1$$

Mathematisch gesprochen bedeutet dieses, d ist die Modulo-Inverse Zahl zu e .

Damit eine Modulo-Inverse zu einer Zahl e existiert muß die Zahl e teilerfremd zu dem Modulus $\phi(n)$ sein. Die Zahl e zum Beispiel kann frei gewählt werden unter der Bedingung, daß sie teilerfremd zum Modulus $\phi(n)$ ist.

Die Methode, mit der jetzt d berechnet werden kann, wird euklidischer Algorithmus genannt.

Name : Namik Ley 496200542	Sicherheit	09.07.1999
		Prof. Dr. T. Risse

Man erhält den Geheimtext c , indem man die Nachricht m mit dem öffentlichen Schlüssel e des Empfängers potenziert und modulo n reduziert:

$$c = m^e \bmod n$$

Der Empfänger entschlüsselt den Geheimtext c , indem er ihn mit seinem geheimen Schlüssel potenziert und modulo n reduziert. Er erhält also die Zahl :

$$m' = c^d \bmod n$$

Die Zahl m' muß gleich der ursprünglichen Nachricht m sein:

$$m' = c^d \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n$$

Nach der Wahl von e und d folgt mit dem Eulerschen Satz

$$m^{ed} \bmod n = m$$

Somit ist $m' = m$.

2.1 Hybridverfahren

Der Weg vom Klartext zum Chiffre und zurück ist mittels RSA, wie dargestellt, recht aufwendig und daher selbst in Hardwarelösungen (im Vergleich zu symmetrischen Verfahren) nicht besonders schnell. Die sogenannten hybriden Verfahren nun versuchen die guten Eigenschaften von symmetrischen und asymmetrischen Verfahren in sich zu vereinen. So ist es zum Beispiel möglich, mit Hilfe des RSA-Verfahrens einen Schlüssel für ein symmetrisches System verschlüsselt zu übertragen, um dann die eigentlichen Daten symmetrisch verschlüsselt übertragen zu können. Schon vor dem RSA-Algorithmus wurde ein Schlüsselaustausch gefunden, bei dem beide Partner gleichberechtigt zusammenarbeiten.

2.2 Anschlußbetrachtung

Die Sicherheit des RSA-Algorithmus beruht auf der Schwierigkeit, die Zahl n wieder in ihre Primfaktoren p und q zu zerlegen. Wenn ein Angreifer in der Lage ist, die Zahl n in ihre Primfaktoren zu zerlegen, befindet er sich in der gleichen Lage wie der Schlüsselerzeuger. Er kann genauso einfach wie dieser den geheimen Schlüssel d berechnen. Daher müssen die Zahlen p und q so gewählt werden daß die Zahl n möglichst schwer zu faktorisieren ist. Insbesondere muß n eine große Zahl sein: in der Praxis werden Zahlen mit mindestens 112 Dezimalstellen gewählt.

Der Clou an diesem Algorithmus liegt im folgendem: Wenn es dem Angreifer gelingt (durch Parallele Datenverarbeitung usw.) in kurzer Zeit die Schlüssel p und q ausfindig zu machen, wird daraufhin die Schlüssellänge auf ein Vielfaches verlängert. Der RSA-Algorithmus ist öffentlich und trotzdem ist es daher so schwierig, diesen zu „knacken“.

Name : Namik Ley 496200542	Sicherheit	09.07.1999
		Prof. Dr. T. Risse

Die Schwierigkeit eine RSA kodierte zu „knacken“ verhält sich exponential gegenüber der Schlüssellänge.

Bei einer Schlüssellänge von 128-Bit sind 100 Million Pentium-Rechner in der Lage, diese Nachricht in ein paar Sekunden zu knacken. Für eine Schlüssellänge von 4096 Bit, benötigt diese Rechenpower ca. 270.000 Jahre, um eine Nachricht zu entschlüsseln.

2.3 Digitale Unterschrift

Die Authentizität von Nachrichten (zum Beispiel Briefen) wird normalerweise durch eine Unterschrift bestätigt. Wesentliche Forderungen an eine digitale Unterschrift, die mit kryptologischen Methoden erzeugt werden kann, sind:

- Nur der rechtmäßige Absender eines Dokuments kann die Unterschrift erzeugen.
- Der Empfänger des Dokuments kann die Unterschrift zweifelsfrei prüfen.
- Die Unterschrift gilt nur im Zusammenhang mit dem gegebenen Dokument.

Bei der digitalen Unterschrift wird ein asymmetrisches Verfahren wie RSA 'rückwärts' angewandt. Der Besitzer des geheimen Schlüssels kann damit genauso gut Nachrichten verschlüsseln, die dann andere mit Hilfe des öffentlichen Schlüssels wieder entschlüsseln können. Somit sind schon alle oben genannten Forderungen an die digitale Unterschrift erfüllt.

Aus der Notwendigkeit, längere Nachrichten in Blöcke aufteilen zu müssen, können bei einer Unterschrift nach diesem Verfahren allerdings die folgenden Probleme auftreten:

- Briefköpfe enthalten zum Beispiel oft den gleichen Text; nur das Datum ändert sich manchmal. Ein Angreifer könnte daher alte, unterschriebene Texte aufzeichnen und somit ein anderes Datum einsetzen, wenn er den ersten Block durch den entsprechenden Block einer älteren Nachricht ersetzt.
- Als Empfänger einer Nachricht weiß ich nicht, ob ich diese komplett empfangen habe; ein Angreifer könnte den letzten Block, der etwa ein Postskriptum enthält, unterdrücken. Um diesen Angriff zu vereiteln, könnte der erste Block die Länge der Nachricht enthalten.

Ein anderes Problem liegt darin, daß das Verschlüsseln einer ganzen Nachricht wegen der Komplexität des RSA-Verfahrens oft zu lange dauert. Eine elegante Lösung dieser Probleme böte eine Funktion, die eine beliebig lange Nachricht im Ganzen auf eine Art 'Fingerabdruck' komprimiert, der in Form einer Zahl vorgegebener Maximallänge angegeben wird.

Diese Funktion kann nicht umkehrbar zu sein (andernfalls wäre sie eine ideale Kompressionsfunktion, die beliebig lange Nachrichten um einen vorgegebenen Faktor komprimiert). Die Funktion ist mit einem Informationsverlust verbunden; aus dem Ergebnis kann in keinem Fall auf die vollständige Nachricht geschlossen

Name : Namik Ley 496200542	Sicherheit	09.07.1999
		Prof. Dr. T. Risse

werden. Es muß ferner nahezu unmöglich sein, zu einem gegebenen Fingerabdruck auch nur irgendeine passende Nachricht zu finden. Falls sich auch nur ein Bit der Nachricht ändert, muß sich ein völlig anderer Fingerabdruck ergeben. Funktionen mit diesen Eigenschaften heißen Hashfunktionen.

Als Unterschrift kann der mit Hilfe des geheimen RSA-Schlüssels verschlüsselte Fingerabdruck dienen. Dazu ist nur eine modulare Exponentiation nötig. Wird die Nachricht mit diesem verschlüsselten Fingerabdruck versendet, so können andere den Fingerabdruck mit Hilfe des öffentlichen Schlüssels entschlüsseln, die gleiche Hashfunktion auf die Nachricht anwenden und beides vergleichen. Als Beispiel dient an dieser Stelle die schon etwas betagte Square-Mod-Hashfunktion; heutzutage übliche Hashfunktionen sind noch weitaus komplexer. Ihre Darstellung würde allerdings den Rahmen sprengen.

2.4 Beispiel einer RSA-Verschlüsselung

- man wählt

$$p=3$$

$$q=5$$

Da es hier nur um ein kleines Beispiel handelt, verwenden wir nur sehr kleine Primzahlen.

- daraus bildet man das Produkt $n=p*q = 3 * 5 = 15$
- $\phi(n)=(p-1)*(q-1) = 2*4 = 8$
- $(e * d) \bmod \phi(n) = 1$

$$\Rightarrow d = e^{-1} \pmod{\phi(n)}$$

1. Bedingung : $e < \phi(n)$
2. Bedingung : e teilerfremd zu $\phi(n)$

$$\Rightarrow e = 3$$

$$\Rightarrow d = 11$$

- public key :

$$n = 15$$

$$e = 3$$

- private key :

$$n = 15$$

Name : Namik Ley 496200542	Sicherheit	09.07.1999
		Prof. Dr. T. Risse

$$d = 11$$

- Sender verschlüsselt die Zahl 13 :

$$c = m^e \bmod n = 13^3 \bmod 15 = 7$$

- Empfänger die empfangende Zahl 7 wieder in Klartext entschlüsseln

$$m = c^d \bmod n = 7^{11} \bmod n = 13$$

Will man ganze Bytefolgen kodieren, darf man dieses auf keinen Fall Byteweise tätigen, da man bei Text, durch Musteranalyse, den Klartext in wenigen Minuten wieder dekodieren kann. Also faßt man anhand der Bedingung $m < e$, so viele Bytes zusammen und kodiert sie dann anschließend. Nun kann man keine Musteranalyse mehr vornehmen.

3.0 Pretty Good Privacy

Die zur Zeit wohl bekannteste Anwendung von IDEA ist Philip Zimmermanns 'Pretty Good Privacy' (PGP; www.pgpi.com), ein verbreitetes Public-Domain-Chiffrierprogramm. PGP verwendet IDEA zur eigentlichen Datenverschlüsselung und RSA für die Schlüsselverwaltung. Selbst ein sonst eher nüchterner Bruce Schneier findet für PGP nur Lob: 'PGP is well designed and nicely coded ... It's the closest you're likely to get to military-grade encryption.' PGP, das sich inzwischen als Standard-'Briefumschlag' für private EMail in öffentlichen Netzen etabliert hat, sorgt bei Staatsschützern für Kopfzerbrechen: Noch nie konnten Informationen gleich welcher Art so gut vor Zugriffen Dritter verborgen werden - und das mit Hilfsmitteln, die für jedermann zugänglich sind.

Leider hat diese Software Auflagen bekommen, die es für Geheimdienste möglich machen, EMail mit verträglichem Aufwand (d.h. in einigen Sekunden, da die Zahlen p und q mit gespeichert werden) wieder in Klartext umzuwandeln.

4.0 Literaturhinweis

- c't 8/94 Seite 230 – Datenschlösser : Grundlagen der Kryptologie
Hagen Hagemann und Andreas Rieke