Das RSA-Kryptosystem

7

Nachdem Whitfield Diffie und Martin Hellman 1976 die asymmetrische Kryptografie eingeführt hatten, tat sich ein neuer Zweig der Kryptologie auf. Diffie und Hellman hatten zwar ein asymmetrisches Schlüsselaustauschverfahren sowie das Konzept der asymmetrischen Verschlüsselung vorgeschlagen, allerdings noch keinen Verschlüsselungsalgorithmus. Daraufhin versuchten Ronald Rivest, Adi Shamir und Leonard Adleman (Abb. 7.1), eine asymmetrische Verschlüsselung zu realisieren, und schlugen 1977 das RSA-Verfahren vor, das das populärste asymmetrische Kryptoverfahren wurde.

In diesem Kapitel erlernen Sie

- die Funktionsweise des RSA-Verfahrens,
- praktische Aspekte von RSA wie z. B. die Berechnung der Parameter und schnelle Ver- und Entschlüsselung,
- Sicherheitsabschätzung von RSA,
- Implementierungsaspekte.

7.1 Einführung

Das RSA-Kryptoverfahren, manchmal auch als das Rivest-Shamir-Adleman-Verfahren bezeichnet, ist das derzeit meistgebrauchte asymmetrische kryptografische Verfahren. Insbesondere für neuere Anwendungen werden allerdings zunehmend elliptische Kurven oder Verfahren basierend auf dem diskreten Logarithmus eingesetzt. RSA war in den USA (nicht aber im Rest der Welt) bis zum Jahr 2000 patentiert.

In der Praxis wird RSA hauptsächlich für die folgenden Aufgaben eingesetzt:

- Verschlüsselung von kleinen Datenmengen, insbesondere für den Schlüsseltransport;
- digitale Signaturen, die in Kap. 10 betrachtet werden, z. B. für Zertifikate im Internet.

© Springer-Verlag Berlin Heidelberg 2016

C. Paar, J. Pelzl, *Kryptografie verständlich*, eXamen.press, DOI 10.1007/978-3-662-49297-0_7

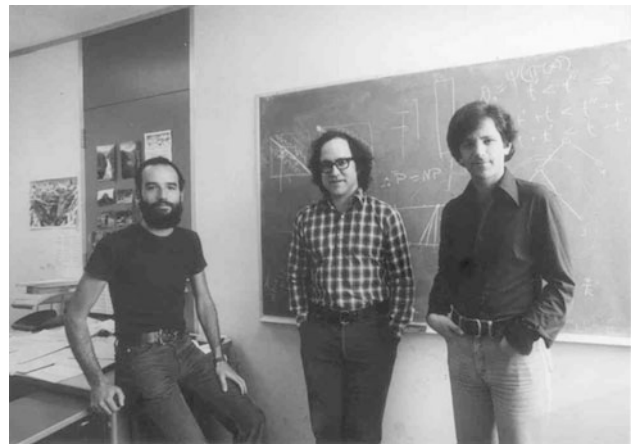

Abb. 7.1 Ein Bild mit den RSA-Erfindern Adi Shamir, Ron Rivest und Leonard Adleman aus den 1970er-Jahren (Wiedergabe mit Erlaubnis von Ron Rivest)

Obwohl man mit dem Algorithmus verschlüsseln kann, löst RSA symmetrische Chiffren nicht ab, da es um ein Vielfaches langsamer als Algorithmen wie AES oder 3DES ist. Dies liegt an dem sehr hohen Rechenaufwand, den RSA (und alle anderen asymmetrischen Algorithmen) erfordern, wie wir später in diesem Kapitel sehen werden. Daher liegt die Hauptanwendung der RSA-Verschlüsselung in dem sicheren Austausch eines Schlüssels – man spricht hier vom Schlüsseltransport – für eine symmetrische Chiffre. In der Praxis wird RSA häufig zusammen mit einem symmetrischen Algorithmus wie AES verwendet, wobei dieser die eigentliche Verschlüsselung der Nutzdaten übernimmt.

Die Einwegfunktion, die RSA zugrunde liegt, ist das Faktorisierungsproblem ganzer Zahlen: Die Multiplikation zweier großer Primzahlen ist rechnerisch einfach (man kann dies selbst für relativ große Zahlen mit Papier und Bleistift durchführen), aber die Faktorisierung des resultierenden Produkts ist sehr schwierig. Der Satz von Euler und die eulersche Phi-Funktion, die in Kap. 6 eingeführt wurden, spielen eine wichtige Rolle im Rahmen des RSA-Verfahrens. Im Folgenden werden wir zunächst beschreiben, wie die Ver- und Entschlüsselung sowie die Schlüsselgenerierung funktionieren, danach werden die praktischen Aspekte von RSA besprochen.

7.2 Ver- und Entschlüsselung

Ver- und Entschlüsselung werden im Restklassenring $\mathbb{Z}_n$ durchgeführt, in dem alle Berechnungen modulo der Zahl n erfolgen. Wir erinnern uns, dass Restklassenringe und die modulare Arithmetik in Abschn. 1.4.2 eingeführt wurden. RSA verschlüsselt Klartexte x, wobei wir die Bits, die x bilden, als Element von $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ betrachten. Folglich muss der binäre Wert des Klartexts x kleiner als n sein. Das Gleiche gilt für das Chiffrat. Verschlüsselung mit dem öffentlichen Schlüssel und Dechiffrierung mit dem privaten Schlüssel erfolgen wie folgt:

RSA-Verschlüsselung
Gegeben seien der öffentliche Schlüssel $(n, e) = k_{\text{pub}}$ und der Klartext x. Die Verschlüsselungsfunktion ist:

$$y = e_{k_{\text{pub}}}(x) \equiv x^e \bmod n \tag{7.1}$$

mit $x, y \in \mathbb{Z}_n$.

RSA-Entschlüsselung
Gegeben seien der private Schlüssel $(d) = k_{\text{pr}}$ und das Chiffrat y. Die Entschlüsselungsfunktion ist:

$$x = d_{k_{\text{pr}}}(y) \equiv y^d \bmod n \tag{7.2}$$

mit $x, y \in \mathbb{Z}_n$.

In der Praxis sind x, y, n und d sehr große Zahlen, üblicherweise mindestens 1024 Bit lang[1]. Der Wert von e wird manchmal auch als *Verschlüsselungsexponent* oder *öffentlicher Exponent* und der private Schlüssel d als *Entschlüsselungsexponent* oder *privater Exponent* bezeichnet. Wenn Alice eine verschlüsselte Nachricht an Bob schicken möchte, benötigt Alice seinen öffentlichen Schlüssel (n, e), und Bob entschlüsselt mit seinem privaten Schlüssel d. In Abschn. 7.3 besprechen wir, wie diese wichtigen Parameter d, e und n erzeugt werden.

Auch ohne alle Details zu kennen, können wir bereits einige Anforderungen an das RSA-Kryptosystem ableiten:

1. Da ein Angreifer Zugriff auf den öffentlichen Schlüssel hat, muss es rechnerisch unmöglich sein, den privaten Schlüssel d bei gegebenen öffentlichen Werten e und n zu berechnen.
2. Da x nur bis zur Größe des Moduls n eindeutig ist, können wir nicht mehr als l Bit mit einer RSA-Verschlüsselung chiffrieren, wobei l die Bitlänge von n ist.
3. Es muss relativ einfach sein, $x^e \bmod n$ und $y^d \bmod n$ zu berechnen, d. h. zu ver- bzw. zu entschlüsseln. Dies bedeutet, dass wir eine effiziente Methode für die schnelle Exponentiation mit sehr großen Zahlen benötigen.
4. Für ein gegebenes n sollten sehr viele Schlüsselpaare existieren, da ansonsten ein Angreifer einen Brute-Force-Angriff durchführen kann. (Es stellt sich heraus, dass diese Anforderung leicht zu erfüllen ist.)

[1] Heutzutage werden Parameter von mindestens 2048 Bit empfohlen, vgl. Tab. 6.1 und die dazugehörige Diskussion.

7.3 Schlüsselerzeugung und Korrektheitsbeweis

Alle asymmetrischen Verfahren haben eine Initialisierungsphase (oder Set-up-Phase), in der der öffentliche und der private Schlüssel berechnet werden. Abhängig von dem asymmetrischen Verfahren kann die Schlüsselerzeugung recht komplex sein. Man beachte, dass die Schlüsselerzeugung für Block- oder Stromchiffren üblicherweise kein Problem ist. Nachfolgend werden die Schritte für die Berechnung des öffentlichen und privaten Schlüssels für das RSA-Kryptosystem gezeigt:

RSA-Schlüsselerzeugung
Ausgabe: Öffentlicher Schlüssel $k_{\text{pub}} = (n, e)$ und privater Schlüssel $k_{\text{pr}} = (d)$

1. Wähle zwei große Primzahlen p und q
2. Berechne $n = p \cdot q$
3. Berechne $\Phi(n) = (p-1)(q-1)$
4. Wähle den öffentlichen Exponenten $e \in \{1, 2, \ldots, \Phi(n) - 1\}$, sodass

$$\text{ggT}(e, \Phi(n)) = 1$$

5. Berechne den privaten Schlüssel d, sodass

$$d \cdot e \equiv 1 \bmod \Phi(n)$$

Die Bedingung $\text{ggT}(e, \Phi(n)) = 1$ stellt sicher, dass die Inverse von e modulo $\Phi(n)$ existiert und es damit auch den privaten Schlüssel d gibt.

Zwei Teile der Schlüsselerzeugung sind nicht trivial: Schritt 1, in dem zwei große Primzahlen gewählt werden, und Schritte 4 und 5, die den öffentlichen und den privaten Schlüssel berechnen. Die Erzeugung der Primzahlen in Schritt 1 ist komplex und wird in Abschn. 7.6 diskutiert. Die Berechnung der Schlüssel d und e kann in einem Schritt mit dem erweiterten euklidischen Algorithmus (EEA) erfolgen. In der Praxis wird häufig zuerst der öffentliche Schlüssel e im Bereich $0 < e < \Phi(n)$ gewählt. Der Wert von e muss die Bedingung $\text{ggT}(e, \Phi(n)) = 1$ erfüllen. Wir wenden dann den EEA mit den Eingabeparametern $\Phi(n)$ und e an und erhalten den Ausdruck:

$$\text{ggT}(\Phi(n), e) = s \cdot \Phi(n) + t \cdot e$$

Wenn $\text{ggT}(e, \Phi(n)) = 1$ ist, folgt, dass e ein gültiger öffentlicher Schlüssel ist. Darüber hinaus wissen wir auch, dass der mit dem EEA berechnete Parameter t die Inverse von e modul $\Phi(n)$ ist und somit:

$$d \equiv t \bmod \Phi(n)$$