Bedeutung von starken Primzahlen für die heutige Zeit

Eine Einführung in das RSA-Verschlüsselungssystem

Winterer, Mathis Aaron

29. Juli 2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Kryptographie und Verschlüsselungssysteme wurden schon im Antiken Rom verwendet um Nachrichten zu verschlüsseln[Aic22], doch heutzutage wären Verschlüsselungssysteme wie der „Cäsar-Chiffre“ gegenüber eines Computers nutzlos. Die Rolle von Kryptographie ist aufgrund des leichten Zugangs der Allgemeinheit zu leistungsstarken Computern, welche das Brechen von „schwachen“ Verschlüsselungssystemen automatisiert und beschleunigt haben, um einiges angestiegen. Aufgrund der nun verfügbaren Rechenleistung wurde die Notwendigkeit für stärkere Verschlüsselungssysteme immer größer. Zur Zeit verwendete Verschlüsselungssysteme sind so konzipiert, dass sie mathematisch schlecht rückwärts zu berechnen sind. Hierzu werden standardmäßig große Primzahlen verwendet, da zur Zeit kein effizienter Algorithmus zur Berechnung von großen Primzahlfaktoren bekannt ist, welcher nicht die Verwendung eines Quantencomputers erfordern würde. Das „brute-forcing“ moderner Verschlüsselungssysteme ist so ineffizient, dass nur ein Teilschritt bis zu fünfzehnhundert Jahre dauern kann[Kle+10]. „Für unsere Berechnungen waren mehr als 10^{20} Operationen erforderlich. Mit dem Äquivalent von fast 2000 Jahren Rechenzeit auf einem AMD Opteron mit einem Kern von 2,2 GHz […]“[Kle+10]. Verschlüsselungssysteme sind aufgrund dieser hohen Anforderungen gegenüber der Resistenz gegen Angriffe äußerst komplexe Systeme welche in der Informationssicherheit eine entscheidende Rolle bei der Sicherung von sensiblen Daten sowohl in der Übertragung als auch bei der Speicherung spielen[Aic22]. Das Ziel dieser Facharbeit ist es den weit verbreiteten und etablierten RSA-Algorithmus zu Analysieren und zu Implementieren. Hierfür werden die einzelnen Teilschritte und verwendeten mathematischen Prinzipien und Funktionen erläutert und beispielhaft dargestellt sowie als Teil eines Programms in C# implementiert, so dass die Funktionen der Schlüsselerzeugung, der Verschlüsselung, der Entschlüsselung und der Signierung verfügbar sind.

1.1 Hintergrund der RSA-Verschlüsselung

$n - 1 = d \times 2^j$

MillerRabin.cs

```cs
private static (BigInteger, BigInteger) FirstPart(BigInteger n)
{
    // nMinusOne
    BigInteger nMo = n - 1;
    BigInteger j = 0;
    BigInteger d = nMo;

    // Divide n − 1 By 2 Until The Result Is Odd, Incrementing `j`
    ↪ Everytime And Storing The Result In `d`, So That `n − 1 = d * 2^j`
    while (d % 2 == 0)
    {
        d /= 2;
        j++;
    }

    return (d, j);
}
```

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Tabelle 1.1: Tabelle

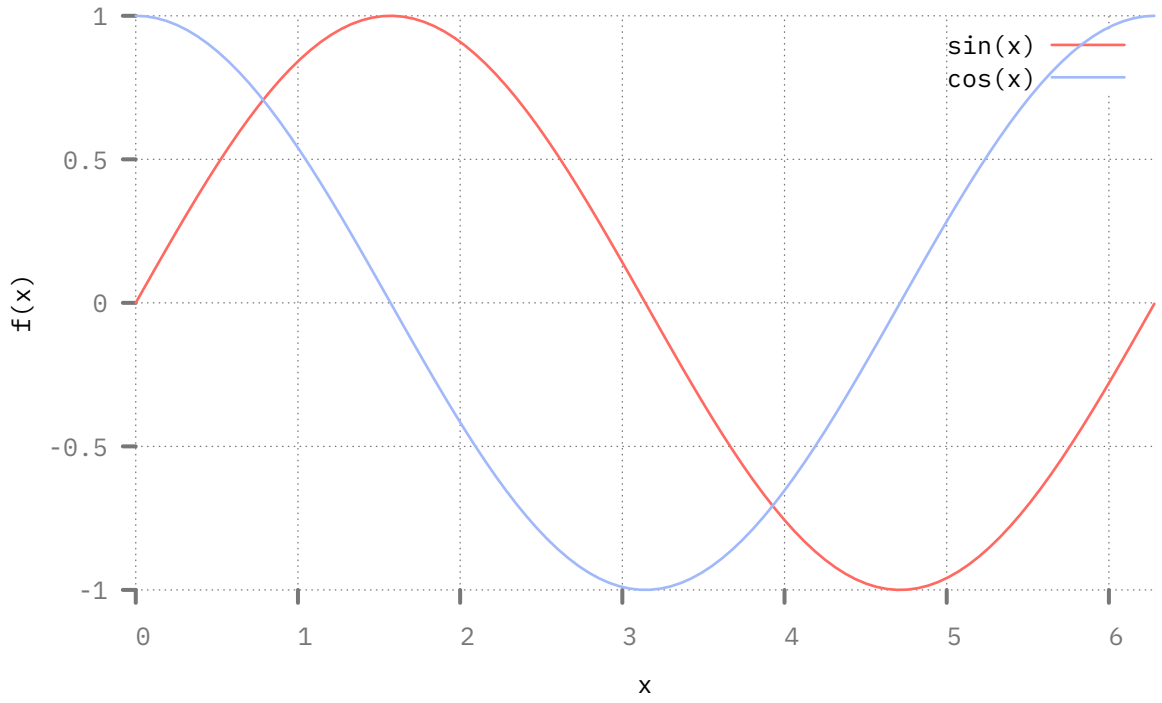

Abbildung 1.1: sin(x) über [0; 6,28]

1.2 Anwendungsbereiche der RSA-Verschlüsselung

2. Mathematische Grundlagen der RSA-Verschlüsselung

2.1 Starke Primzahlen

2.2 Primzahltests

2.2.1 Miller-Rabin-Test

2.2.2 Kleiner fermatscher Satz

2.2.2.1 Fermatscher Primzahltest

2.3 Eulersche Phi-Funktion

2.4 Carmichael-Funktion

2.5 Euklidischer Algorithmus

2.5.1 Erweiterter Euklidischer Algorithmus

2.6 Hamming-Gewicht

2.7 Chinesischer Restsatz

3. Anwendungsbeispiel

3.1 Schlüsselerzeugung

3.2 Voraussetzungen

Für die Erzeugung eines Schlüsselpaares werden zwei zufällige große Primzahlen p und q so gewählt, dass eine Fastprimzahl $n = pq$ berechnet werden kann. Im Verlauf dieser Arbeit wird immer angenommen, dass p und q die selbe Bitgröße besitzen $pbit = qbit$, wobei dies für den RSA-Algorithmus nicht zwingend notwendig ist. Um das Errechnen von p und q durch Faktorisierung von n zu Erschweren sollten Primzahlen für p und q gewählt werden, welche beide „groß“[1] und „weit“ auseinander liegen. Beide Primfaktoren p und q sind durch die Bitgröße von n, für welche in der Regel eine Zweierpotenz gewählt wird, beschränkt, wobei $pbit$ und $qbit$ in der Regel entweder gleich groß oder annähernd gleich groß sind. $pbit$ und $qbit$ seien für jeden Wert p und q als die Bitgröße dieser definiert. Für $p \geq 2$ sei $q \geq 3 \land q \neq 2$ und für $p \geq 3$ sei $q \geq 2 \land q \neq 3$, da dies die kleinsten Primzahlen sind für welche ein n berechnet werden kann:

$$\begin{aligned}
&\{\, p \in \mathbb{P} \mid (q = 2 \implies 3 \leq p) \oplus (q = 3 \implies (2 \leq p \land p \neq 3)) \,\} \\
&\{\, q \in \mathbb{P} \mid (p = 2 \implies 3 \leq q) \oplus (p = 3 \implies (2 \leq q \land q \neq 3)) \,\} \\
&\{\, pbit \in \mathbb{N} \mid 2 \leq pbit \,\} \\
&\{\, qbit \in \mathbb{N} \mid 2 \leq qbit \,\} \\
&pbit = \left\lfloor \frac{\ln p}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 p \rfloor + 1 \\
&qbit = \left\lfloor \frac{\ln q}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 q \rfloor + 1
\end{aligned} \tag{3.1}$$

Somit sei für eine Zahl $p \coloneqq 2^{2048} - 1$ die Bitgröße $pbit$ dementsprechend:

$$pbit = \lfloor \log_2 p \rfloor + 1 = 2048 \tag{3.2}$$

Folgend sei für jeden Wert n die Bitgröße $nbit$:

$$\begin{aligned}
&\{\, nbit \in \mathbb{N} \mid 3 \leq nbit \,\} \\
&nbit = \left\lfloor \frac{\ln n}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 n \rfloor + 1 \\
&nbit = pbit + qbit
\end{aligned} \tag{3.3}$$

Somit ergibt sich für $nbit \coloneqq 4096$, $pbit \coloneqq 2048$ und $qbit \coloneqq 2048$:

$$\left\{ n \in \mathbb{P} \times \mathbb{P} \mid n \neq 4 \land 6 \leq n \leq 2^{4096} - 1 \right\} \tag{3.4}$$

[1] Derzeitig sichere RSA-Schlüssel besitzen normalerweise zwischen 1024- und 4096-bit

Zwischen 1991 und 2007 veröffentlichte RSA Laboratories die sogenannten „RSA-Zahlen“. Diese Zahlen waren verschiedene Werte n zwischen 100- und 2048-bit. Ziel dieser Aktion war es die Forschung im Bereich der effizienten Faktorisierung voranzutreiben, dies wurde durch beträchtliche Preisgelder für einige dieser Zahlen bezweckt. Zum derzeitigem Standpunkt ist „RSA-250“ die zuletzt Faktorisierte „RSA-Zahl“ mit 829-bit[Zim20]. Die nächste „RSA-Zahl“ „RSA-260“ enthält 862-bit und wurde bis zu diesem Zeitpunkt nicht faktorisiert. Somit ergeben sich für p, q, n und $nbit$ durch die Primfaktoren von „RSA-250“ neue Mindestanforderungen:

$$\begin{aligned}
a &= 6413528947707158027879019017057738908482501474294344720811685963202453234463023862359875266834770873766192558569463979885336 7 \\
&\hookrightarrow \quad 5323446302386235987526683477087376619255856946397988533 67 \\
b &= 3337202759497815655622601060535511422794076034476755466678452098702 3 \\
&\hookrightarrow \quad 8417292100370802574486732968818775657189862580369320627 11 \\
&\{ p \in \mathbb{P} \mid (p = a \implies a \leq p \land b \leq q) \oplus (p = b \implies b \leq p \land a \leq q) \} \\
&\{ q \in \mathbb{P} \mid (q = a \implies a \leq q \land b \leq p) \oplus (q = b \implies b \leq q \land a \leq p) \} \\
&\left\{ n \in \mathbb{P} \times \mathbb{P} \mid 2^{862} - 1 < n \right\} \\
&\{ nbit \in \mathbb{N} \mid 862 < nbit \}
\end{aligned} \tag{3.5}$$

Das Bundesamt für Sicherheit in der Informationstechnik empfiehlt die Verwendung von RSA-Schlüsseln mit mehr als 3000-bit, um sicherzustellen, dass der Modulus n derzeit nicht faktorisiert werden kann.[Bun24] Somit ergeben sich erneut neue Definitionen für p, q, n, $pbit$, $qbit$ und $nbit$:

$$\begin{aligned}
&\left\{ p \in \mathbb{P} \mid 2^{1500} - 1 \leq p \right\} \\
&\left\{ q \in \mathbb{P} \mid 2^{1500} - 1 \leq q \right\} \\
&\left\{ n \in \mathbb{P} \times \mathbb{P} \mid 2^{3000} - 1 \leq n \right\} \\
&pbit = \left\lfloor \frac{\ln p}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 p \rfloor + 1 = 1500 \\
&qbit = \left\lfloor \frac{\ln q}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 q \rfloor + 1 = 1500 \\
&nbit = \left\lfloor \frac{\ln n}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 n \rfloor + 1 = 3000 \\
&nbit = pbit + qbit \\
&n = p \times q
\end{aligned} \tag{3.6}$$

Als Teil meiner Arbeit werde ich $nbit = 4096$ als Bitgröße verwenden, da 4096-Bit die derzeit größte weit verbreitete Schlüsselgröße bei RSA-Schlüsseln ist. Im Weiteren werde ich davon ausgehen, dass p und q die selbe Bitröße besitzen. Somit ergeben sich für diese Arbeit folgende finale Definitionen für p, q, n, $pbit$, $qbit$ und $nbit$:

$$
\begin{aligned}
&\left\{ p \in \mathbb{P} \mid p \leq 2^{2048} - 1 \right\} \\
&\left\{ q \in \mathbb{P} \mid p \leq 2^{2048} - 1 \right\} \\
&\left\{ p \in \mathbb{P} \mid (p = 2 \implies (2 \leq p \leq 2^{2048} - 1 \land p \neq 3) \land (3 \leq q \leq 2^{2048} - 1)) \oplus \right. \\
&\left. \hookrightarrow (p = 3 \implies (3 \leq p \leq 2^{2048} - 1) \land (2 \leq q \leq 2^{2048} - 1 \land q \neq 3)) \right\} \\
&\left\{ q \in \mathbb{P} \mid (q = 2 \implies (2 \leq q \leq 2^{2048} - 1 \land q \neq 3) \land (3 \leq p \leq 2^{2048} - 1) \oplus \right. \\
&\left. \hookrightarrow (q = 3 \implies (3 \leq q \leq 2^{2048} - 1) \land (2 \leq p \leq 2^{2048} - 1 \land p \neq 3)) \right\} \\
&\left\{ n \in \mathbb{P} \times \mathbb{P} \mid n \leq 2^{4096} - 1 \right\} \\
&pbit = 2048 \\
&qbit = 2048 \\
&nbit = 4096
\end{aligned}
\tag{3.7}
$$

3.3 Schlüsselverteilung

3.4 Verschlüsselung

3.5 Entschlüsselung

3.6 Signatur

4. Auswertung

4.1 Faktorisierung von „RSA-155“

Als Teil meiner Arbeit und als Demonstration der Nötigkeit von immer größeren Bitgrößen für n habe ich unter Verwendung der „CADO-NFS“ Software, welche eine Implementation des „**N**umber **F**ield **Sieve**“-Algorithmuses (Zahlkörpersiebalgorithmus) bereitstellt eine Primfaktorzerlegung vorgenommen. Für Zahlen größer als 10^{100} ist der Zahlkörpersieb der effizienteste bekannte Algorithmus zur Primfaktorzerlegung[Har]. „RSA-155“ ist eine Fastprimzahl mit 512-bit und somit größer als 10^{100}, welche 1999 von einem Team unter der Leitung von Herman te Riele in acht Monaten faktorisiert wurde[Cav+99]. Fünfundzwanzig Jahre später war ich in der Lage „RSA-155“ auf einem Intel Pentium G4400T aus dem Jahre 2015 in nur rund zwanzig Tagen[1] zu faktorisieren. Dies zeigt klar, dass die Faktorisierung von Fastprimzahlen im Laufe der Zeit durch Fortschritte im Bereich der Primfaktorzerlegung und einen breiteren öffentlichen Zugang zu Leistungsstarken Computern und entsprechenden Softwarepaketen um einiges erleichtert wurde. Dadurch werden größere Werte n zwingend nötig, da RSA-Schlüssel sonst in für einen Angreifer realisierbarer Zeit gebrochen werden könnten. Angreifer mit ausreichenden Rechenresourcen, wie zum Beispiel staatlich unterstützte Akteure oder solche mit Cloud- oder Clustercomputern können RSA-Schlüssel mit solch geringer Bitgröße in einer noch kürzeren Zeit brechen. Meine Faktorisierung hat die erwarteten Primfaktoren 10660348838016845482092722036001287867920795857598929152227060823719306280864 3 und 10263959282974110577205419657399167590071656780803806680334193352179071130777 9 geliefert, welche auch 1999 so errechnet wurden:

$$
\begin{aligned}
p &= 106603488380168454820927220360012878679207958575989291522270608237193062808643 \\
q &= 102639592829741105772054196573991675900716567808038066803341933521790711307779 \\
n = pq &= 109417386415705274218097073220403576120037329454492059909138421314763499 8 \\
&\hookrightarrow \quad 428893478471799725789126733249762575289978183379707653724402714674353159335433389 7
\end{aligned}
\tag{4.1}
$$

[1] 1.77025×10^6 Sekunden entsprechen 20 Tagen 11 Stunden 44 Minuten und 08 Sekunden

4.2 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel „Bedeutung von starken Primzahlen – Eine Einführung in das RSA-Verschlüsselungssystem“ selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die den verwendeten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

_____

Ort, Datum Unterschrift

Literatur

[Aic22] Daniela Aichner. „Primzahlen und ihre Bedeutung in der Kryptographie“. Seiten 45, 48. Diplomarbeit. Innsbruck, Österreich: Universität Innsbruck, Feb. 2022. URL: https://www.uibk.ac.at/mathematik/algebra/media/teaching/diplomarbeit.pdf (besucht am 28.06.2024).

[Bun24] Bundesamt für Sicherheit in der Informationstechnik. *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. Technische Richtlinie BSI TR-02102-1. Seite 35. Bundesamt für Sicherheit in der Informationstechnik, Feb. 2024. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf (besucht am 09.07.2024).

[Cav+99] Stefania Cavallar u.a. *Factorization of a 512-bit RSA Modulus*. Aug. 1999. URL: https://www.iacr.org/archive/eurocrypt2000/1807/18070001-new.pdf (besucht am 29.07.2024).

[con24a] Wikipedia contributors. *Fermat primality test — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/w/index.php?title=Fermat_primality_test%5C&oldid=1227031378 (besucht am 28.06.2024).

[con24b] Wikipedia contributors. *Miller–Rabin primality test — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/w/index.php?title=Miller%E2%80%93Rabin_primality_test%5C&oldid=1222212331 (besucht am 28.06.2024).

[con24c] Wikipedia contributors. *RSA (cryptosystem) — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/w/index.php?title=RSA_(cryptosystem)%5C&oldid=1221958777 (besucht am 28.06.2024).

[Har] Sam Harwell. *What is the fastest integer factorization algorithm?* [Archivierter Link]. URL: https://stackoverflow.com/a/2274520 (besucht am 29.07.2024).

[Kle+10] Thorsen Kleinjung u.a. *Factorization of a 768-Bit RSA modulus*. Feb. 2010. URL: https://eprint.iacr.org/2010/006.pdf (besucht am 28.06.2024).

[Rob99] Eric Roberts. *Prime Generation & Testing for primality*. 1999. URL: https://www-cs-faculty.stanford.edu/people/eroberts/courses/soco/projects/1998-99/randomized-algorithms/applications/primality.html (besucht am 28.06.2024).

[RSA01] R. L. Rivest, A. Shamir und L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Apr. 2001. URL: https://people.csail.mit.edu/rivest/Rsapaper.pdf (besucht am 04.07.2024).

[Sha49] C. E. Shannon. „Communication theory of secrecy systems“. In: *The Bell System Technical Journal* 28.4 (1949), S. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x. URL: https://ieeexplore.ieee.org/document/6769090 (besucht am 28.06.2024).

[Zim20] Paul Zimmermann. *Factorization of RSA-250*. Feb. 2020. URL: https://sympa.inria.fr/sympa/arc/cado-nfs/2020-02/msg00001.html (besucht am 04.07.2024).