Bedeutung von starken Primzahlen für die heutige Zeit

Eine Einführung in das RSA-Verschlüsselungssystem

Winterer, Mathis Aaron

10. 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

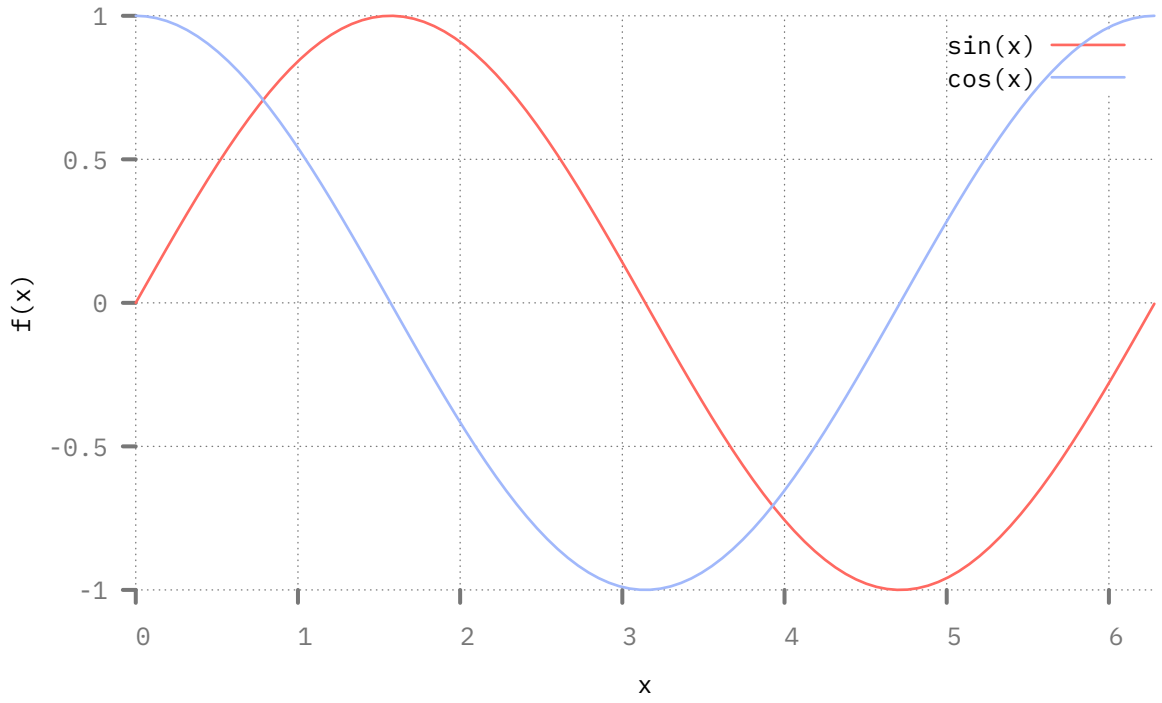

Abbildung 1.1: sin(x) über [0; 6,28]

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

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, 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 p und q 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 $p \coloneqq 2$ sei $q \coloneqq 3$ und für $p \coloneqq 3$ sei $q \coloneqq 2$, da dies die kleinsten Primzahlen sind für welche ein funktionierendes n berechnet werden kann:

$$\{ p \in \mathbb{P} \mid (q = 2 \implies 3 \leq p) \oplus (q = 3 \implies 2 \leq p) \} \tag{3.1}$$

$$\{ q \in \mathbb{P} \mid (p = 2 \implies 3 \leq q) \oplus (p = 3 \implies 2 \leq q) \} \tag{3.2}$$

$$\{ pbit \in \mathbb{N} \mid 2 \leq pbit \} \tag{3.3}$$

$$\{ qbit \in \mathbb{N} \mid 2 \leq qbit \} \tag{3.4}$$

$$pbit = \left\lfloor \frac{\ln p}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 p \rfloor + 1 \tag{3.5}$$

$$qbit = \left\lfloor \frac{\ln q}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 q \rfloor + 1 \tag{3.6}$$

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

$$pbit = \lfloor \log_2 p \rfloor + 1 = 2048 \tag{3.7}$$

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

$$\{ nbit \in \mathbb{N} \mid 3 \leq nbit \} \tag{3.8}$$

$$nbit = \left\lfloor \frac{\ln n}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 n \rfloor + 1 \tag{3.9}$$

$$nbit = pbit + qbit \tag{3.10}$$

So ergeben sich für $nbit \coloneqq 4096$ p, q und n:

$$\left\{ n \in \mathbb{P}^2 \mid 6 \leq n \leq 2^{4096} - 1 \right\} \tag{3.11}$$

[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:

$$a=641352894770715802787901901705773890848250147429434472081168596320245323446302386235987526683477087376619255856946397988533 67 \tag{3.12}$$

$$b=333720275949781565562260106053551142279407603447675546667845209870238417292100370802574486732968818775657189862580369320627 11 \tag{3.13}$$

$$\{p \in \mathbb{P} \mid (p = a \implies a \leq p \wedge b \leq q) \oplus (p = b \implies b \leq p \wedge a \leq q)\} \tag{3.14}$$

$$\{q \in \mathbb{P} \mid (q = a \implies a \leq q \wedge b \leq p) \oplus (q = b \implies b \leq q \wedge a \leq p)\} \tag{3.15}$$

$$\{n \in \mathbb{P}^2 \mid 2^{862} - 1 < n\} \tag{3.16}$$

$$\{nbit \in \mathbb{N} \mid 862 < nbit\} \tag{3.17}$$

Das deutsche Bundesamt für Sicherheit in der Informationstechnik empfiehlt die Verwendung von RSA-Schlüsseln mit mehr als 3000-Bit.[Bun24] Somit ergeben sich erneut neue Definitionen für p, q, n, $pbit$, $qbit$ und $nbit$:

$$\{p \in \mathbb{P} \mid 2^{1500} - 1 \leq p\} \tag{3.18}$$

$$\{q \in \mathbb{P} \mid 2^{1500} - 1 \leq q\} \tag{3.19}$$

$$\{n \in \mathbb{P}^2 \mid 2^{3000} - 1 \leq n\} \tag{3.20}$$

$$pbit = \left\lfloor \frac{\ln p}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 p \rfloor + 1 = 1500 \tag{3.21}$$

$$qbit = \left\lfloor \frac{\ln q}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 q \rfloor + 1 = 1500 \tag{3.22}$$

$$nbit = \left\lfloor \frac{\ln n}{\ln 2} \right\rfloor + 1 = \lfloor \log_2 n \rfloor + 1 = 3000 \tag{3.23}$$

$$nbit = pbit + qbit \tag{3.24}$$

$$n = p \times q \tag{3.25}$$

Als Teil meiner Arbeit werde ich $nbit = 4096$ als Bitgröße verwenden. 4096-Bit ist die derzeit größe verwendete Schlüsselgröße bei RSA-Schlüsseln.

3.2 Schlüsselverteilung

3.3 Verschlüsselung

3.4 Entschlüsselung

3.5 Signatur

Literatur

[Aic22] Daniela Aichner. „Primzahlen und ihre Bedeutung in der Kryptographie“. [Online; accessed 28-June-2024]. Diplomarbeit. Innsbruck, Österreich: Universität Innsbruck, Feb. 2022. URL: https://www.uibk.ac.at/mathematik/algebra/media/teaching/diplomarbeit.pdf.

[Bun24] Bundesamt für Sicherheit in der Informationstechnik. *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. Technische Richtlinie BSI TR-02102-1. [Online; accessed 09-07-2024]. 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.

[con24a] Wikipedia contributors. *Fermat primality test — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Fermat_primality_test%5C&oldid=1227031378.

[con24b] Wikipedia contributors. *Miller–Rabin primality test — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Miller%E2%80%93Rabin_primality_test%5C&oldid=1222212331.

[con24c] Wikipedia contributors. *RSA (cryptosystem) — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=RSA_(cryptosystem)%5C&oldid=1221958777.

[Kle+10] Thorsen Kleinjung u.a. *Factorization of a 768-Bit RSA modulus*. [Online; accessed 28-June-2024]. Feb. 2010. URL: https://eprint.iacr.org/2010/006.pdf.

[Rob99] Eric Roberts. *Prime Generation & Testing for primality*. [Online; accessed 28-June-2024]. 1999. URL: https://www-cs-faculty.stanford.edu/people/eroberts/courses/soco/projects/1998-99/randomized-algorithms/applications/primality.html.

[RSA01] R. L. Rivest, A. Shamir und L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. [Online; accessed 04-July-2024]. Apr. 2001. URL: https://people.csail.mit.edu/rivest/Rsapaper.pdf.

[Sha49] C. E. Shannon. „Communication theory of secrecy systems“. In: *The Bell System Technical Journal* 28.4 (1949). [Online; accessed 28-June-2024], S. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x. URL: https://ieeexplore.ieee.org/document/6769090.

[Zim20] Paul Zimmermann. *Factorization of RSA-250*. [Online; accessed 04-July-2024]. Feb. 2020. URL: https://sympa.inria.fr/sympa/arc/cado-nfs/2020-02/msg00001.html.