Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung
Dr. Helmut Reiser

MNM TEAM
Munich Network Management Team

IT-Sicherheit im Wintersemester 2008/2009
Übungsblatt 5

Abgabetermin: 26.11.2008 bis 14:00 Uhr

Achtung: Die schriftlichen Lösungen aller mit H gekennzeichneten Aufgaben sind **vor Beginn** der jeweils nächsten Übungsveranstaltung abzugeben. Während des Semesters werden drei Übungsblätter korrigiert. Bei drei richtigen Lösungen erfolgt ein Bonus von zwei drittel Notenstufen auf die Klausurnote, bei zwei richtigen Lösungen erhalten Sie einen Notenbonus von einer drittel Notenstufe.

Aufgabe 10: (H) RSA

a. Berechnen Sie den öffentlichen und privaten Schlüssel für das RSA Verfahren basierend auf den Primzahlen 7 und 11. Wählen Sie als Wert für den Verschlüsselungsexponenten 23.

b. Verschlüsseln Sie mit Hilfe des RSA Verfahrens und dem oben berechneten öffentlichen Schlüssel die Zahl 6.

c. Entschlüsseln Sie ihre Nachricht mit Hilfe des RSA Verfahrens und dem oben berechneten privaten Schlüssel.

Aufgabe 11: (K) Schlüssellängen

a. Wie lange dauert es mit einem gegebenen Rechner (3GHz, ca. $3 * 10^6 \frac{\text{Schlüssel}}{\text{s}}$) einen symmetrischen Schlüssel der Länge 56 Bit / 128 Bit mittels Brute Force zu brechen?

b. Wie lange benötigt man, um mit der genannten Maschine einen 4096 Bit langes RSA Modul zu brechen?

c. Wie lange benötigt im Vergleich die Copacobana (20 Module ´a 120 FPGAs, $27.000.000 \frac{\text{Schlüssel}}{\text{s} * \text{FPGA}}$) für 56 Bit / 128 Bit lange DES Schlüssel?

d. Wie würde sich die Existenz eines DNA- oder Quantencomputers auf die Sicherheit von DES, AES und RSA auswirken?

Aufgabe 12: (K) Miller-Rabin Primzahltest

a. Wie lange benötigt ein handelsüblicher Rechner mittels Probedivision, um eine 58 Bit lange natürliche Zahl als Primzahl zu identifizieren, wenn er 1000 Divisionen pro Sekunde durchführen kann?

b. Der Miller-Rabin Primzahltest arbeitet deutlich schneller, ist aber ein probabilistisches Verfahren und gehört daher in die Klasse der Monte-Carlo Algorithmen. Der Algorithmus arbeitet folgendermaßen:

(i) Teste ob die zu prüfende Zahl n gerade ist. Falls ja ist n zu 100% keine Primzahl.

(ii) Anderenfalls: Berechne n-1 und schreibe die Zahl als $u * 2^k$, so dass u ungerade.

(iii) Wähle zufällig ein a aus N, so dass $1 < a < n - 1$

(iv) Test1: $a^u \mod n = 1$

(v) Test2: Existiert ein i aus N mit $0 < i < k$, so dass $a^{u*2^i} \mod n = n - 1$?

(vi) Falls sowohl Test1 als auch Test2 negativ sind, ist n zu 100% keine Primzahl, falls wenigstens einer der beiden Tests positiv ausfällt, steigt die Wahrscheinlichkeit, dass n eine Primzahl ist auf $1 - (\frac{1}{4})^{\#\mathrm{Durchläufe}}$

(vii) Wiederhole das Verfahren ab Punkt biii.

Prüfen Sie mit Hilfe des Miller-Rabin Primzahltests, ob 49 und 29 Primzahlen sind. Stellen Sie sicher, dass ihr Ergebniss mit 85%-iger Wahrscheinlichkeit korrekt ist.

c. Implementieren Sie den Miller-Rabin Primzahltest als Java-Programm! Testen Sie mit ihrem Programm ob 6465454655446847 eine Primzahl ist.