FTK3, WS 2023/24
2. Übungsblatt für den 25.10.2023

Die öffentlichen Schlüssel für die Beispiele 2 und 4 befinden sich in den beiliegenden Textfiles.

1. Alices öffentlicher RSA-Schlüssel ist $(308911, 87943)$. Du weißt, dass Alice einen kleinen privaten Exponenten d verwendet. Bestimme Alices privaten Schlüssel mit der Attacke von Wiener und erkläre deine Lösung Schritt für Schritt.

2. Bob verwendet auf seinem Yubikey einen 4096-Bit-RSA-Schlüssel mit kleinem privaten Exponenten d, um schnell entschlüsseln zu können. Zeige Bob, dass das keine gute Idee ist, indem du mit der Attacke von Wiener und der Programmiersprache deiner Wahl[1] sowohl seinen privaten Exponenten als auch die Primfaktoren von n aus seinem öffentlichen Schlüssel (n, e) ermittelst. Wie kannst du überprüfen, dass die gefundene Lösung korrekt ist?

3. Carols öffentlicher RSA-Schlüssel ist $(3718548079, 65537)$. Du weißt, dass Carols Primfaktoren nah beieinander liegen. Bestimme Carols privaten Schlüssel, indem du n mit der Fermat-Methode faktorisierst. Erkläre deine Lösung Schritt für Schritt.

4. Dan verwendet auf seinem Canon-Drucker einen 4096-Bit-RSA-Schlüssel mit Primfaktoren, die nah beieinander liegen. Bestimme Dans privaten Schlüssel, indem du n mit der Fermat-Methode und der Programmiersprache deiner Wahl faktorisierst. Wie kannst du überprüfen, ob die gefundene Lösung korrekt ist?

Betrachte in den Beispielen 5 bis 7 die Gruppe[2] $(G, \circ, \widehat{\ }, \perp)$. Zähle alle Elemente von G auf. Für jedes Element g schreibe eine Liste mit g, g^2, g^3, g^4, etc., bis das Ergebnis $\perp$ ist. Markiere in der Liste jeweils das Element $\widehat{g}$ und bestimme die Ordnung von g.

5. $(G, \circ, \widehat{\ }, \perp) = (\mathbb{Z}_{10}, +, -, 0)$
6. $(G, \circ, \widehat{\ }, \perp) = (\mathbb{Z}_{11}^*, \cdot, ^{-1}, 1)$
7. $(G, \circ, \widehat{\ }, \perp) = (\mathbb{Z}_{15}^*, \cdot, ^{-1}, 1)$
8. Sieh dir nochmal genau die Gruppen aus Beispiel 2.3 im Vorlesungsskriptum an. Nenne nun analog dazu jeweils zwei Gruppen der Ordnung
   (a) 30
   (b) 20
   (c) 1

[1] Wenn du dich für Python entscheidest, können die Funktionen `continued_fraction`, `cf_approx` und `cf_approx_from_cf` aus dem Modul `si.py` und `isqrt` aus dem Modul `math` hilfreich sein.

[2] Schlage zur Erinnerung die Beispiele 0 und 1 des 13. GDK-Übungszettels nach.