FTK3, WS 2023/24
6. Übungsblatt für den 21.12.2023

Implementiere das Lamport-Signaturverfahren für 1-Byte-Nachrichten in der Programmiersprache deiner Wahl.

1. Implementiere eine Funktion `GenerateKeys`, die Private Key und Public Key retourniert.
2. Implementiere eine Funktion `Sign`, die eine übergebene Nachricht mit einem übergebenen Private Key signiert.
3. Implementiere eine Funktion `Verify`, die mit Hilfe des übergebenen Public Keys verifiziert, ob die übergebene Signatur zur übergebenen Nachricht passt.

Du hast eine Winternitz-Signatur für die Nachricht `"Ich hasse dich"` (kodiert in Latin-1) vorliegen, bei der auf die Prüfbits c_0, c_1, c_2 [1] am Ende verzichtet wurde.

4. Kannst du daraus ohne Kenntnis des Private Keys eine Signatur für die Nachricht `"Ich küsse dich"` erzeugen? Wenn ja, wie? Wenn nein, wieso nicht?
5. Kannst du daraus ohne Kenntnis des Private Keys eine Signatur für die Nachricht `"Ich liebe dich"` erzeugen? Wenn ja, wie? Wenn nein, wieso nicht?

Im File `wots.py` findest du eine Python-Implementierung von Winternitz-One-Time-Signaturen ohne Prüfsumme. Im File `signature` findest du eine Signatur der Nachricht

```
Sende 1000 Euro an P0000000000000000@lab.fh-hagenberg.at
```

6. Erzeuge mit Hilfe der `sign`-Funktion daraus eine Signatur der Nachricht

   `Sende 9999 Euro an S2210239xxx@students.fh-hagenberg.at`
   (Verwende deine eigene E-Mail-Adresse!)
7. Prüfe deine Signatur mit dem Public Key aus dem File `pubkey`.

Das File `wots.py` hat also WOTS nicht korrekt implementiert.

8. Erweitere die Implementierung um die Berechnung der Prüfsumme (also der 12 Prüfbits c_0, c_1, c_2), die mitsigniert wird. Die Prüfsumme soll sowohl in der `sign`- als auch in der `verify`-Funktion berechnet werden.

[1] siehe Seite 71 im Vorlesungsskriptum