Aufgabe 1 ECDSA-Signatur mit bekannter Nonce oder Nonce Reuse

1) Wir wissen bereits, dass die Signatur s mit der Formel $k^{-1}(h(m)+d\cdot r) \mod q$ berechnet wird. Stellt man diese um, so kommt man sehr leicht auf den geheimen Schlüssel d:

$$\begin{aligned} s = k^{-1}(h(m) + d \cdot r) &\iff s \cdot k = h(m) + d \cdot r \\ &\iff s \cdot k - h(m) = d \cdot r \\ &\iff (s \cdot k - h(m)) \cdot r^{-1} = d \end{aligned}$$

Dadurch, dass sich das multiplikativ inverse von r sehr effizient berechnen lässt, kann der private Schlüssel d sehr einfach berechnet werden, sobald man k kennt.

2) Hier kann man einfach überprüfen, ob $r_1 = r_2$ gilt. Sobald das der Fall ist, wurde zwangsläufig ein und das selbe k benutzt.

3) Hier muss man wieder ein wenig umstellen:

$$\begin{aligned} s_1 - s_2 &= k^{-1}(h(m_1) + d \cdot r) - k^{-1}(h(m_2) + d \cdot r) \\ &= k^{-1}(h(m_1) + \cancel{d \cdot r} - h(m_2) - \cancel{d \cdot r}) \\ &= k^{-1}(h(m_1) - h(m_2)) \\ \iff (s_1 - s_2) \cdot (h(m_1) - h(m_2))^{-1} &= k^{-1} \end{aligned}$$

Somit kann man sich sehr leicht ein k^{-1} errechnen. Bildet man davon das multiplikative Inverse, so ist man im Besitz von k und hat somit die selben Voraussetzungen wie im ersten Aufgabenteil. Mit der dort gezeigten Formel kann man sich dann den privaten Schlüssel errechnen.

4) Der private Schlüssel lautet

```
729356443913076564618991391779299508262542616843769625200832389017
36736586421
```

5) Der Klartext lautet

```
ITS_KoSyS{Reuse_your_nonce_just_once_and_we_know_your_key}%
```

Aufgabe 2 Kurze Vektoren in Lattices

1) Hier kann man das Theorem von Hermite benutzen, um eine obere Schranke für den kürzesten Vektor zu berechnen. Setzen wir unsere Werte dort ein, so kommen wir auf den folgenden Wert:

$$\begin{aligned} n^{\frac{n}{2}} \cdot det(L) &= 251^{\frac{251}{2}} \cdot 2^{2251.58} \\ &\geq ||\vec{v_1}|| \cdot ||\vec{v_2}|| \cdot \ldots \cdot ||\vec{v_n}|| \end{aligned}$$

Somit wurde erst einmal eine obere Schranke das Produkt aller Vektoren definiert. Zieht davon nun die 251te Wurzel von dem Ergebnis, so hat man eine obere Schranke für den kürzesten Vektor. In diesem Fall ist dies $\approx 7497,08$. Wären alle Vektoren gleich lang, so wäre das die Länge des kürzesten Vektors und damit auch aller Vektoren. Trifft das nicht zu, so kann es durchaus auch kürzere Vektoren geben.

2) Das Prinzip dahinter ist ziemlich simpel. Da wir uns im zweidimensionalen Raum befindet, gibt es auch nur zwei Basisvektoren. In dem Verfahren wird jetzt immer von dem jeweils längeren Vektor eine bestimmte Skalierung des kürzeren Vektors abgezogen. Diese Skalierung berechnet sich nach einer bestimmten, wohldefinierten Formel. Ist der vorher kürzere Vektor nach einer Iteration länger, so werden die beiden Vektoren getauscht und das Verfahren wiederholt. Im Falle eines Latice wird die vorher errechnete Skalierung noch kaufmännisch gerundet, bevor der kürzere Vektor damit multipliziert, also herunter skaliert, wird.

Aufgabe 3 Babai's Algorithmus und Lösen des CVP mit einer "guten" Basis

1) Der Sinn von Babai's Algorithmus ist es, das Closest Vector Problem zu lösen. Wir wollen also zu einem beliebigen Punkt, der sich nicht in dem Latice befindet, einen möglichst nahen Latice-Punkt finden. Der Algorithmus geht dabei so vor, dass er sich eine Basis des Latice nimmt, und den Punkt als Linearkombination der Basisvektoren darstellt. Dadurch, dass die sich daraus ergebenden Koeffizienten reell und nicht ganzzahlig sind (ansonsten wäre der gegebene Punkt nämlich schon ein Latice-Punkt), werden diese anschließend noch gerundet. Somit ergibt sich ein Latice-Punkt, welcher sich möglichst nah an dem gegebenen Punkt befindet. Das Verfahren steht und fällt natürlich mit der genutzten Basis. Je "besser" die Basis, desto näher ist der Latice-Punkt an dem Zielpunkt.
2) Der Code befindet sich in der `sheet2_template.py` Datei.
3) Die *Hadamard ratio* von $(\vec{v_1}, \vec{v_2})$ ist ziemlich hoch, nämlich ≈ 0.967. Damit ist es eine "gute" Basis, da die *Hadamard ratio* aufschluss auf die Orthogonalität gibt. Je orthogonaler die Matrix ist, desto höher ist auch die *Hadamard ratio* und umso "besser" ist auch die Basis.
4) Der Rechenweg zu dieser Aufgabe befindet sich ebenfalls in der `sheet2_template.py` Datei. Ich löste das Problem mittels des Gauss-Verfahrens und bekam so eine Basiswechselmatrix:

$$A = \begin{bmatrix} 5 & 19 \\ 6 & 23 \end{bmatrix}$$

Diese Matrix ist offensichtlich ganzzahlig. Die Determinante ist genau 1.

5) $\mathcal{H}(\{\vec{v_1'}, \vec{v_2'}\}) \approx 0.047$. Nach der oben genannten Argumentation ist das eine ziemlich "schlechte" Basis.

Aufgabe 4 Kyber

Aufgabe 5 LLL-Algorithmus

1) Der Code befindet sich in der `sheet2_template.py` Datei.
2) $det(L) = -21242880806, \mathcal{H}(M) \approx 0.457$. Laut dem Programm ist der kürzeste Vektor $\vec{v_6}$ mit einer Länge von ≈ 63.198.
3)

$$M^{LLL} = \begin{bmatrix} -6 & -3 & -2 & 2 & -26 & 10 \\ 11 & 30 & 2 & 5 & -6 & 24 \\ -14 & -10 & 14 & -48 & -3 & -6 \\ -3 & 24 & 43 & 23 & -33 & -38 \\ 64 & -44 & -16 & -46 & -13 & 4 \\ -28 & -25 & 41 & 5 & 30 & 39 \end{bmatrix}$$

- Der *swap*-Schritt wird 11 mal ausgeführt.
- $\mathcal{H}(M^{LLL}) \approx 0.934$

4)

$$M'^{LLL} = \begin{bmatrix} 6 & 3 & 2 & -2 & 26 & -10 \\ 11 & 30 & 2 & 5 & -6 & 24 \\ 14 & 10 & -14 & 48 & 3 & 6 \\ -28 & -25 & 41 & 5 & 30 & 39 \\ -3 & 24 & 43 & 23 & -33 & -38 \\ 47 & -35 & 54 & 30 & -13 & 11 \end{bmatrix}$$

- Der *swap*-Schritt wird 8 mal ausgeführt.
- $\mathcal{H}(M'^{LLL}) \approx 0.944$

5)

$$M^{LLL} = \begin{bmatrix} -6 & -3 & -2 & 2 & -26 & 10 \\ 11 & 30 & 2 & 5 & -6 & 24 \\ -14 & -10 & 14 & -48 & -3 & -6 \\ -3 & 24 & 43 & 23 & -33 & -38 \\ -28 & -25 & 41 & 5 & 30 & 39 \\ 47 & -35 & 54 & 30 & -13 & 11 \end{bmatrix}$$

- Der *swap*-Schritt wird 12 mal ausgeführt.
- $\mathcal{H}(M^{LLL}) \approx 0.944$