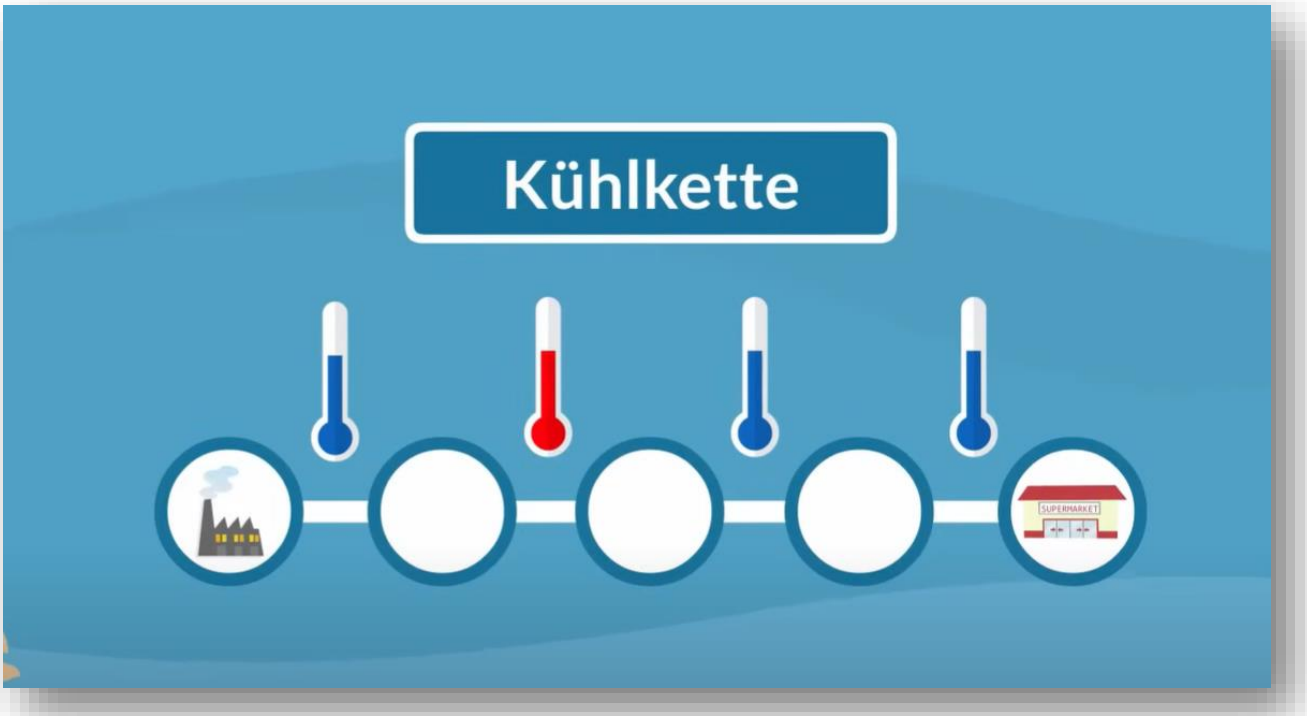
IoT-Kühlkettenüberwachung

ETS-Supplychain-Projekt

- Phase 2 -



Inhalt

1 IoT-Kühlkettenüberwachung – Projektstufe 2

1.1 Aufgabenstellung

Der Hersteller „Food Solution Hildesheim“ möchte die Kühlkettenüberwachung aus Projektstufe 1 durch drei Funktionen ergänzen:

1. Temperaturüberwachung der Kühlstationen
2. Lieferdatenverschlüsselung
3. Wetterdatenabfrage an den Auslagerorten

2 Organisatorische Rahmenbedingungen

- Bearbeiten Sie die Aufgaben in einer Gruppe, die aus **sechs Personen** besteht.
  Jedes Gruppenmitglied soll eigenständige Programmieraufgaben übernehmen!
- Nutzen Sie die Programmiersprache **Python**.
  Die drei Erweiterungsaufgaben sollen jeweils in einer eigenen „Funktion“ bearbeitet werden.
  Die drei Funktionen werden in einem Hauptprogramm zusammengeführt.
- Nutzen Sie die Entwicklungsumgebung **Visual-Studio-Code**.
- Synchronisieren Sie das Projekt (von Beginn an) mit **Github**.
  Dokumentieren Sie das Projekt auf Github.
  Gewähren Sie den Lehrkräften Zugriff auf das Github-Projekt.
- Nutzen Sie eine Projektmanagementmethode ihrer Wahl**.**
  Benennen Sie im Rahmen der Projektmanagementmethode verschiedene Rollen in Ihrer Gruppe, insbesondere auch für die Kommunikation mit den Lehrkräften.
- Stellen Sie Ihr Programm in einem 10-minütigen **Vortrag** vor.
  Erklären Sie hier den Programmaufbau, geben Sie einen Überblick über den Programmcode und heben Sie einzelne Besonderheiten hervor. Reflektieren Sie das Projekt abschließend. Achten Sie auf die korrekte Fachsprache und einen systematischen Vortragsaufbau.

3 Bewertung

Die Bewertung erfolgt als Note für die gesamte Projektgruppe.

Bereich	Anteil	Kriterien
Vortrag	20%	• Präsentationskriterien
Programmcode	30%	• Programmaufbau • Programmcode • Funktion
Programmdokumentation	20%	• Rahmendaten (Datum, Verfasser, …) • Programmcode (ev. SPHINX, DOXYGEN, …) • Algorithmus, Grundideen, …
Github	10%	• Dokumentation • Sync-Timeline
Projektmanagemen	20%	• Aufbau und Qualität der Dokumentation • Planung und Steuerung des Projekts • Kommunikation

4 Technische Beschreibung

4.1 Temperaturüberwachung der Kühlstationen

Scenario:

Es hat sich herausgestellt, dass trotz der Kühlkettenüberwachung mehrfach minderwertige Ware ausgeliefert wurde, da die Kühltemperaturvorgaben auf dem Transport nicht eingehalten wurden. Zur Qualitätssicherung sollen zukünftig die Temperaturen der Kühlhäuser und der Transportfahrzeuge ausgewertet werden.

Aufgabe:

Ergänzen Sie Ihr Programm aus Projektstufe 1 so, dass die Einhaltung der Temperaturgrenzwerte in den Kühlstationen überprüft wird.

Technische Spezifikation:

- Jede Kühlstation (Güterverteilzentrum und Kühltransporter) schreibt im zeitlichen Abstand von 15 min Temperaturdaten in die Tabelle „tempdata“:
- Die **Kühltemperaturen** dürfen den Bereich zwischen +2°C und +4°C nicht verlassen.

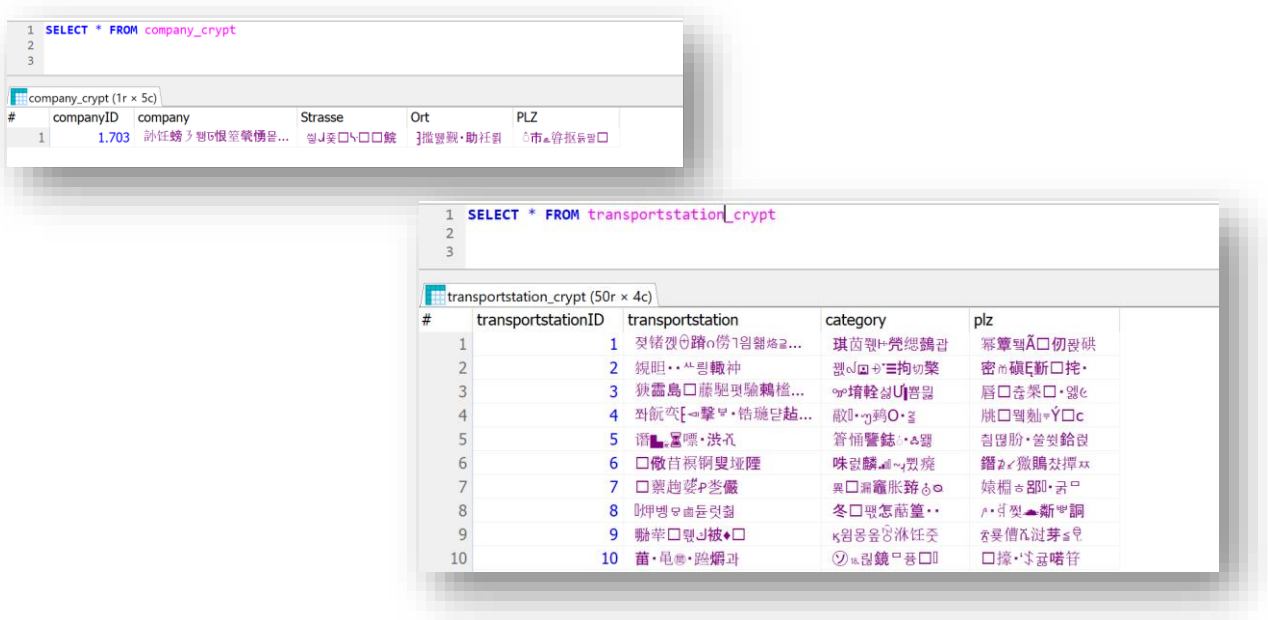
supplychain.tempdata: 734 Zeilen gesamt

#	transportstationID	datetime	temperature
1	2	2022-09-07 06:00:00.000	3,6
2	2	2022-09-07 06:15:00.000	3,2
3	2	2022-09-07 06:30:00.000	3,3
4	2	2022-09-07 06:45:00.000	3,3
5	2	2022-09-07 07:00:00.000	3,4
6	2	2022-09-07 07:15:00.000	3,4
7	2	2022-09-13 12:00:00.000	3,6
8	2	2022-09-13 12:15:00.000	3,6
9	2	2022-09-13 12:30:00.000	3,5
10	2	2022-09-13 12:45:00.000	3,5
11	2	2022-09-13 13:00:00.000	3,6
12	2	2022-09-13 13:15:00.000	3,6
13	2	2022-09-13 13:30:00.000	3,5
14	2	2022-09-13 13:45:00.000	3,5
15	2	2022-09-13 14:00:00.000	3,6
16	2	2024-03-07 19:21:06.000	3,3
17	4	2022-09-20 06:00:00.000	3,4
18	4	2022-09-20 06:15:00.000	3,3
19	4	2022-09-20 06:30:00.000	3,3
20	4	2022-09-20 06:45:00.000	3,4

4.2 Lieferdatenverschlüsselung

Scenario:

Es hat Beschwerden von Kunden gegeben, dass die Lieferdaten öffentlich einsehbar seien, da sie unverschlüsselt abgespeichert werden. Das hat die Firma „Food Solution Hildesheim“ inzwischen geändert und die Daten verschlüsselt in den Tabellen „company_crypt“ und „transportstation_crypt“ abgelegt.



Aufgabe:

Ändern Sie Ihr Programm aus Projektstufe 1 so ab, dass die verschlüsselten Daten verarbeitet werden können.

Technische Spezifikation:

Der Datenbankeinträge sind folgendermaßen verschlüsselt:

- AES-CBC-Verschlüsselung mit Python-Pycryptodome
- Passwort: `b'mysecretpassword'`
- Initialization-Vector: `b'passwort-salzen!'`

4.3 Wetterdatenabfrage an den Auslagerorten

Scenario:

In der Projektstufe 1 durften die Zeiträume ohne Kühlung, also die Zeit zwischen dem Aus- und Einchecken den Maximalwert von 10 min nicht überschreiten. Für alle gefunden Zeitüberschreitungen soll zukünftig zusätzlich die Temperatur am Übergabeort ausgewiesen werden. So kann die Ware ggf. doch für andere Zwecke weiterverwendet werden.

Aufgabe:

Geben Sie für jeden gefunden ***„Zeitraum ohne Kühlung“*** aus der Projektstufe 1 zusätzlich die aktuelle Temperatur am Auslagerungsort zur Auslagerungszeit an.

Technische Spezifikation:

In der Tabelle „transportstation“ ist die Postleitzahl für jede Transportstation vermerkt.

Da die Transportwagen keinem Ort zugeordnet werden können, ist die Postleitzahl 0 eingetragen.

supplychain.transportstation: 50 Zeilen gesamt

#	transportstationID	transportstation	category	plz
1	1	'Antwerp-Trans Waage...	'KT'	0
2	2	'BHL Wagen 126'	'KT'	0
3	3	'Dansk Logistik Varevo...	'KT'	0
4	4	'Darmstadt Local Logist...	'KT'	0
5	5	'DHL Wagen 126'	'KT'	0
6	6	'DHL Wagen 337'	'KT'	0
7	7	'DHL_KT_1033'	'KT'	0
8	8	'DHL_KT_133'	'KT'	0
9	9	'DHL_KT_223'	'KT'	0
10	10	'DHL_KT_256'	'KT'	0
11	11	'DHL_KT_266'	'KT'	0
12	12	'DHL_KT_334'	'KT'	0
13	13	'DHL_KT_339'	'KT'	0
14	14	'DHL_KT_441'	'KT'	0
15	15	'DHL_KT_445'	'KT'	0
16	16	'DHL_KT_456'	'KT'	0
17	17	'DHL_KT_582'	'KT'	0
18	18	'DHL_KT_612'	'KT'	0
19	19	'DHL_KT_792'	'KT'	0
20	20	'FedEx Wagen 005'	'KT'	0
21	21	'GVZ Bielefeld_gibt_es_...	'GVZ'	0
22	22	'GVZ Bremen Kühlhaus 3'	'GVZ'	28201
23	23	'GVZ Dresden Kühlhaus...	'GVZ'	01099
24	24	'GVZ HH Sued Gebäude...	'GVZ'	20146
25	25	'GVZ Kühlhaus 3'	'GVZ'	23560
26	26	'GVZ Lübeck Kühlhaus 1'	'GVZ'	23560
27	27	'GVZ München Area 13'	'GVZ'	80337
28	28	'GVZ Osnabrück Kühlha...	'GVZ'	49082
29	29	'GVZ-Hildesheim-Kühlh...	'GVZ'	31141
30	30	'GVZ-Hildesheim-Kühlh...	'GVZ'	31141
31	31	'GVZ-Hildesheim-Kühlh...	'GVZ'	31141

Sie können später auch die Tabelle „transportstation_crypt“ verwenden!

5 Datenmodell

5.1 Tabellen

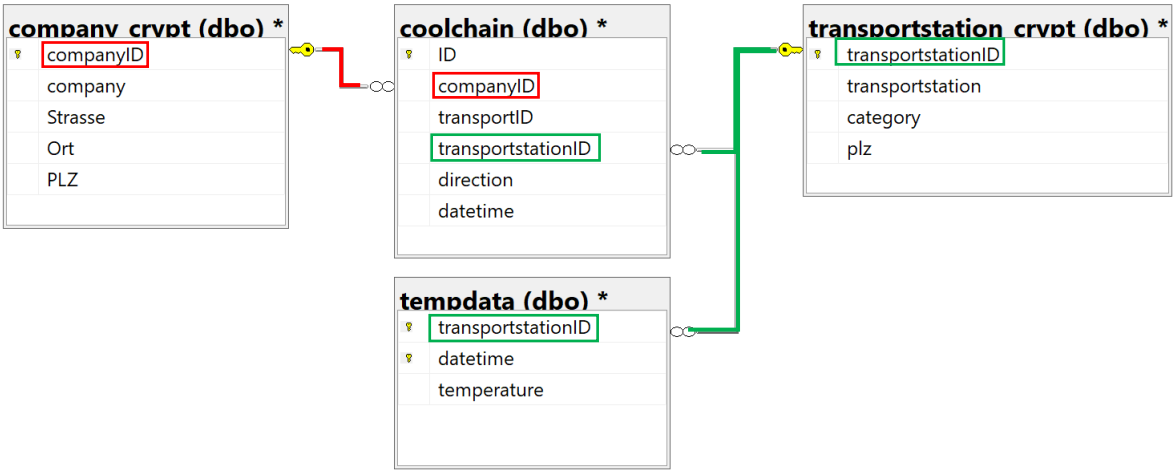
Im Zuge der zweiten Projektstufe wurde die bisherige Tabelle „normalisiert“ und in das folgende Datenmodell überführt. Es besteht aus fünf Tabellen, die über IDs und Fremdschlüssel verbunden sind:

- Tabelle “coolchain“: Transportstationen der Lieferungen
- Tabelle “transportstation_crypt“: Stammdaten der Kühlhäuser (verschlüsselt)
- Tabelle “company_crypt“: Stammdaten der Firmen (verschlüsselt)
- Tabelle “tempdata“: Kühlhaus-Temperaturdaten

Im Rahmen der Normalisierung wurden die Daten in **Stammdaten** und **Bewegungsdaten** getrennt.

In den Stammdaten gibt es für jedes Objekt genau einen Eintrag, der mit einer eindeutigen ID als Primärschlüssel versehen wird. Hierzu gehören die Tabellen „company_crypt“ und „transportstation_crypt“.

In den Bewegungsdaten „coolchain“ und „tempdata“ werden die IDs der Stammdaten als Fremdschlüssel verwendet.



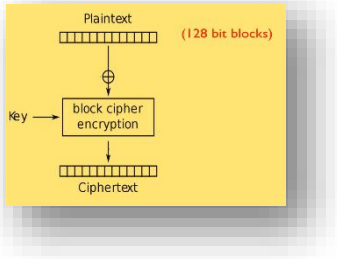
6 Daten verschlüsseln und entschlüsseln

In diesem Beispiel soll ein Text verschlüsselt werden. Hierfür benutzen wir das Verschlüsselungsverfahren Advanced Encryption Standard (AES).

6.1 Symmetrische Verschlüsselung mit AES

AES ist ein symmetrisches Verschlüsselungsverfahren, das einen gemeinsamen Schlüssel für die Verschlüsselung und Entschlüsselung von Daten verwendet. Es funktioniert folgendermaßen:

1. **Schlüsselaustausch**: Bevor Daten gesendet werden, müssen die beteiligten Parteien einen gemeinsamen Schlüssel vereinbaren. Dieser Schlüssel muss sicher übertragen werden, um sicherzustellen, dass keine unbefugte dritte Partei Zugang zu ihm erhält.
2. **Verschlüsselung**: Sobald der Schlüssel ausgetauscht wurde, kann die versendende Partei die Daten mithilfe des AES-Algorithmus und des Schlüssels verschlüsseln. Der AES-Algorithmus teilt die Daten in Blöcke (128 Bit) auf und verschlüsselt jeden Block mit dem Schlüssel.
3. **Übertragung**: Die verschlüsselten Daten werden über eine sichere Verbindung an die empfangende Partei gesendet.
4. **Entschlüsselung**: Die empfangende Partei kann die Daten mithilfe desselben AES-Algorithmus und des gleichen Schlüssels, den sie von der versendenden Partei erhalten hat, entschlüsseln.

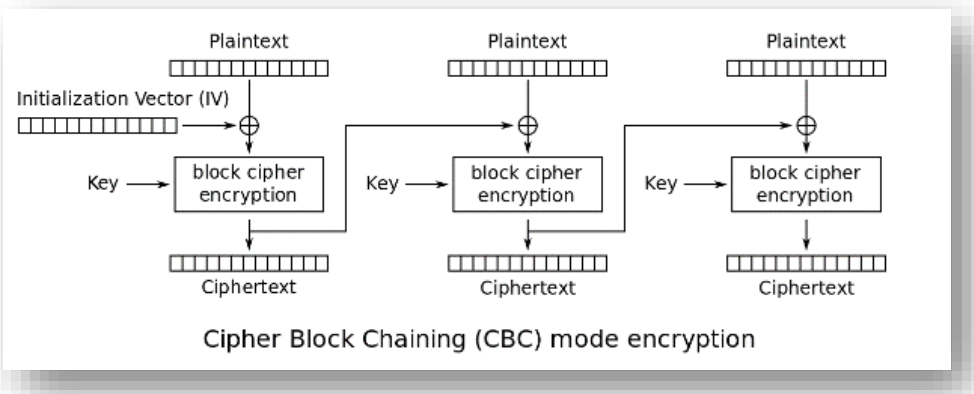


6.2 Passwörter Salzen

Da der AES-Algorithmus öffentlich bekannt ist, gibt es auch bekannte Angriffsszenarien, um das Passwort zu „knacken“. Wenn man z.B. den Kontext eines verschlüsselten Textes kennt und daher vermutet, dass ein bestimmtes Wort besonders häufig vorkommt (z.B. „BZTG“), kann man den verschlüsselten Text auf Wiederholungen analysieren und durch ausprobieren das Passwort ermitteln.

Um dies zu verhindern „salzt“ man die Passwörter. Für den ersten zu verschlüsselnden Block kombiniert man das Passwort mit einem zusätzlichen Wert, dem „Initialization Vektor“. Für alle weiteren Blöcke nimmt man als Salz das verschlüsselte Ergebnis des letzten Blocks.

So ergibt das gleiche Klartextwort (z.B. „BZTG“) mit dem gleichen Passwort, durch das Hinzufügen verschiedener „Salze“, niemals das gleiche Verschlüsselungsergebnis. Damit ist es unmöglich das Passwort zurückrechnen, indem man den verschlüsselten Text auf Wiederholungen analysiert.



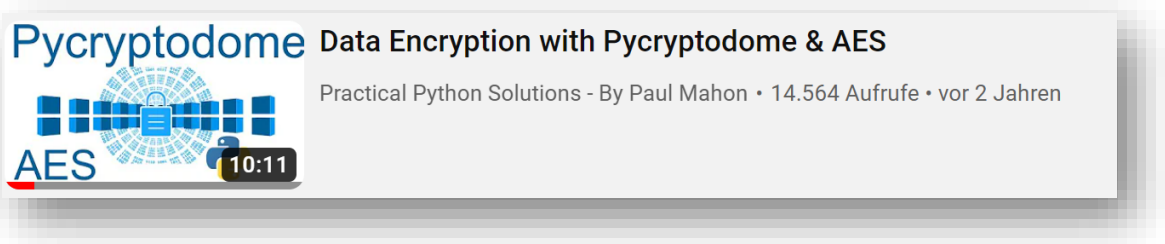
6.3 Padding

Padding bezieht sich auf den Prozess, bei dem Daten auf eine bestimmte Länge angepasst werden, um sie für eine sichere Verschlüsselung geeignet zu machen.

Im Kontext der AES-Verschlüsselung (Advanced Encryption Standard) müssen die Daten, die verschlüsselt werden sollen, in Blöcke von 128 Bit aufgeteilt werden, und es kann vorkommen, dass ein Block nicht die vollständige Größe hat. In diesem Fall müssen die Daten mit Nullen oder anderen Zeichen gefüllt werden, um die vollständige Blockgröße zu erreichen. Dies wird als „Padding“ bezeichnet.

Es ist wichtig zu beachten, dass das Padding nach der Verschlüsselung entfernt werden muss, um die ursprünglichen Daten wiederherzustellen. Dies nennt man „Unpadding“.

Zur Einführung können Sie folgendes Video ansehen: https://www.youtube.com/watch?v=AUxHDlC6Rn0



6.4 Die Bibliothek „pycryptodome“

Pycryptodome ist eine drittanbieter-Bibliothek für Kryptografie in Python. Es ist eine unabhängige Implementierung der Standardbibliothek Python Cryptography Toolkit (PyCrypto) mit aktualisierten Algorithmen und Sicherheitsfunktionen. Pycryptodome bietet eine Vielzahl von Kryptografie-Algorithmen wie AES, DES, RSA und viele andere, die zum Schutz von Daten und zur Authentifizierung von Benutzern verwendet werden können. Es ist ein sehr praktisches Tool für Entwickler, die Kryptografie in ihren Anwendungen einsetzen möchten.

Installieren Sie die Bibliothek „pycryptodome“: `py -m pip install pycryptodome`

6.5 Einen Text verschlüsseln

```
# Bibliotheken
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

# Initialisierung
key = b'mysecretpassword'                # 16 Byte Passwort
iv  = b'passwort-salzen!'                # 16 Byte Initialization Vektor
cipher = AES.new(key, AES.MODE_CBC, iv)  # Verschlüsselung initialisieren

# Verschlüsselung
plaintext = b'Die ist mein Klartext!'  # Klartext
ciphertext = cipher.encrypt(pad(plaintext,AES.block_size))  # Text verschlüsseln

# Ausgabe
print ('-------------------------------------------------------------------------------------------')
print ("Verschlüsselter Text : ", ciphertext)
print ('-------------------------------------------------------------------------------------------')
```

Ausgabe:

```
-------------------------------------------------------------------------------------------
Verschlüsselter Text :  b'\xe0\xdc*\x841\x87;p\xd22\xd9\x94\xabH6\xcd\xf0&\xeduO\x19\x17$+K*wke\x81\xdf'
-------------------------------------------------------------------------------------------
```

6.6 Einen verschlüsselten Text wieder entschlüsseln

Als verschlüsselter Text (ciphertext) wird die Ausgabe des vorherigen Programms verwendet.

```
# Bibliotheken
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

# Initialisierung
key = b'mysecretpassword'               # 16 Byte Passwort
iv  = b'passwort-salzen!'               # 16 Byte Initialization Vektor
cipher = AES.new(key, AES.MODE_CBC, iv)  # Verschlüsselung initialisieren

# Entschlüsselung
ciphertext = b'\xe0\xdc*\x84l\x87;p\xd22\xd9\x94\xabH6\xcd\xf0&\xeduO\x19\x17$+K*wke\x81\xdf'
plaintext = unpad(cipher.decrypt(ciphertext), AES.block_size)  # Text entschlüsseln

# Ausgabe
print ('--------------------------------------------------------------------------')
print ("Entschlüsselter Text als Bytewert: ", plaintext)
print ("Entschlüsselter Text als String:   ", plaintext.decode())
print ('--------------------------------------------------------------------------')
```

Ausgabe:

```
--------------------------------------------------------------------------
Entschlüsselter Text als Bytewert:  b'Die ist mein Klartext!'
Entschlüsselter Text als String:    Die ist mein Klartext!
--------------------------------------------------------------------------
```

6.7 Datenbankabfrage entschlüsseln

In dem folgenden Programm werden die Daten der Tabelle „company_crypt“ entchlüsselt.

```
import pyodbc
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

# Initialisierung
key = b'mysecretpassword'                # 16 Byte Passwort
iv = b'passwort-salzen!'                 # 16 Byte Initialization Vektor
cipher = AES.new(key, AES.MODE_CBC, iv)  # Verschlüsselung initialisieren

# Entschlüsselungsfunktion
def decrypt_value(encrypted_data):
    return unpad(cipher.decrypt(encrypted_data), AES.block_size).decode()

# Verbindungsdaten
server = 'sc-db-server.database.windows.net'
database = 'supplychain'
username = 'rse'
password = 'Pa$$w0rd'

# Verbindungsstring
conn_str = (
    f'DRIVER={{ODBC Driver 17 for SQL Server}};'
    f'SERVER={server};'
    f'DATABASE={database};'
    f'UID={username};'
    f'PWD={password}'
)

# Verbindung herstellen
conn = pyodbc.connect(conn_str)
cursor = conn.cursor()

# Datensätze auslesen
select_query = 'SELECT companyID, company, strasse, ort, plz FROM company_crypt'
cursor.execute(select_query)

# Für jeden Datensatz die Entschlüsselung durchführen und ausgeben
for row in cursor.fetchall():
   companyID, encrypted_company, encrypted_strasse, encrypted_ort, encrypted_plz = row

   # Da die Daten als binär gespeichert wurden, sollte hier keine Umwandlung mit str() erfolgen
   decrypted_company = decrypt_value(encrypted_company)
   decrypted_strasse = decrypt_value(encrypted_strasse)
   decrypted_ort = decrypt_value(encrypted_ort)
   decrypted_plz = decrypt_value(encrypted_plz)

   print(f"ID: {companyID}, Company: {decrypted_company}, Strasse: {decrypted_strasse}, Ort: {decrypted_ort},
PLZ: {decrypted_plz}")

# Verbindung schließen
cursor.close()
conn.close()
```

7 Wetterdaten abfragen

Mit der Wetter-API von „VisualCrossing“ können historische Temperaturdaten an einem spezifischen Datum und einer spezifischen Uhrzeit für einen bestimmten Ort ermittelt werden.

1. **Registrierung**
   Für die Nutzung des Dienstes benötigen Sie einen (kostenlosen) API-Schlüssel.
   Registrieren Sie sich auf https://www.visualcrossing.com/ und kopieren Sie den „key“ aus ihrem „Account“. Fügen Sie den Key später in den Programmcode ein.

2. **Bibliotheken**
   Installieren Sie die Bibliothek „requests“.


```
import requests
import json
from datetime import datetime

# Beispiel-Nutzung
api_key = "Hier_bitte_Ihren_API-KEY_einfügen"
location = "26127,DE"
datetime_str = "10.07.2023 13:00"  # Zeit auf die nächste volle Stunde gerundet

# Konvertiere das Datum und die Uhrzeit in das erforderliche Format
datetime_obj = datetime.strptime(datetime_str, '%d.%m.%Y %H:%M')
timestamp = datetime_obj.strftime('%Y-%m-%dT%H:%M:%S')

# Visual Crossing Weather API-Endpunkt
url =
'https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timelin
e/{location}/{timestamp}'.format(location=location, timestamp=timestamp)
response = requests.get(url, params={'unitGroup': 'metric','key':
api_key,'include': 'hours'})
data = response.json()

# Ausgabe der Temperatur
print("\nTemperatur: ", data["days"][0]["temp"],"\n")

# Ausgabe des gesamten JSON-Objekts
#json_str = json.dumps(data, indent=4)
#print(json_str)
```

Ausgabe:

```
Temperatur:  20.3
```