```python
# todo Dersin Adı: Kriptoloji ||
# todo öğrenci ismi: Doğukan öztürk | Numara: 2205314017 | 2.Sınıf Bilişim Güvenliği Teknolojisi


import binascii  # Binary verileri ASCII formatına dönüştürmek için bir kütüphane
from tkinter import *  # Grafik arayüz bileşenlerini içeren bir kütüphane

import colorama
from Crypto.Cipher import AES, PKCS1_OAEP  # Şifreleme algoritmaları
from Crypto.Random import get_random_bytes  # Rastgele byte'lar üretmek için kullanılan bir fonksiyon
from Crypto.Hash import HMAC, SHA256  # Hash tabanlı mesaj doğrulama kodu algoritmaları
from Crypto.Util.Padding import pad, unpad  # Şifreleme için padding işlemleri
from Crypto.PublicKey import RSA  # RSA şifreleme algoritması
from Crypto.Protocol.KDF import PBKDF2  # Anahtar türetme fonksiyonu
import time
import numpy as np
from colorama import Fore, Style
import random

# Global değişkenler
key = None  # RSA anahtar çifti
encrypted_aes_key = None  # Şifrelenmiş AES anahtarı
encrypted_hmac_key = None  # Şifrelenmiş HMAC anahtarı
ciphertext = None  # Şifrelenmiş mesaj
mac = None  # Mesaj doğrulama kodu
cipher_aes = None  # AES şifreleme nesnesi
original_message = None  # Orijinal mesaj
n_value = None  # RSA n değeri
e_value = None  # RSA e değeri
salt = None  # Tuz değeri
uyari_mesaji = None # Açıklama bölümü
```

```python
# Kullanıcı arayüzü için fonksiyonlar
def bekle_ve_tamamlandi_yaz(textbox):
    for i in np.arange(1, 100, 0.009):  # 1'den 100'e kadar 0.01 adımlarla ondalık sayılarla bir aralık oluştur
        textbox.delete("1.0", END)
        if i <= 50:
            if i <= 10:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color1")
                textbox.tag_config("color1", foreground="blue")
            elif 10 < i <= 20:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color2")
                textbox.tag_config("color2", foreground="green")
            elif 20 < i <= 30:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color3")
                textbox.tag_config("color3", foreground="orange")
            elif 30 < i <= 40:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color4")
                textbox.tag_config("color4", foreground="red")
            else:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color5")
                textbox.tag_config("color5", foreground="purple")
        else:
            if i <= 60:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color6")
                textbox.tag_config("color6", foreground="pink")
            elif 60 < i <= 70:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color7")
                textbox.tag_config("color7", foreground="brown")
            elif 70 < i <= 80:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color8")
                textbox.tag_config("color8", foreground="cyan")
```

```
            else:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color9")
                textbox.tag_config("color9", foreground="magenta")

        textbox.update()
        time.sleep(0.0000)

    time.sleep(1)

    for i in np.arange(99, 0, -0.009):  # 99'dan 0'a kadar 0.01 adımlarla ondalık sayılarla bir aralık oluştur
        textbox.delete("1.0", END)
        if i <= 50:
            if i <= 10:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color1")
                textbox.tag_config("color1", foreground="blue")
            elif 10 < i <= 20:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color2")
                textbox.tag_config("color2", foreground="green")
            elif 20 < i <= 30:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color3")
                textbox.tag_config("color3", foreground="orange")
            elif 30 < i <= 40:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color4")
                textbox.tag_config("color4", foreground="red")
            else:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color5")
                textbox.tag_config("color5", foreground="purple")
        else:
            if i <= 60:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color6")
                textbox.tag_config("color6", foreground="pink")
            elif 60 < i <= 70:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color7")
                textbox.tag_config("color7", foreground="brown")
            elif 70 < i <= 80:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color8")
                textbox.tag_config("color8", foreground="cyan")
            else:
                textbox.insert(END, f"Bekleyin... %{i:.2f}\n", "color9")
                textbox.tag_config("color9", foreground="magenta")

            textbox.update()
            time.sleep(0.0000)

        textbox.delete("1.0", END)
        textbox.insert(END, "Tamamlandı!\n", "tamamlandi")
        textbox.tag_config("tamamlandi", foreground="green")
        textbox.update()
        time.sleep(1)


            # Şifreleme fonksiyonu
    def sifrele():
        # Global değişkenlerin tanımlanması
        # Bu değişkenler, şifreleme ve deşifreleme işlemleri sırasında kullanılacak anahtarlar, metinler ve diğer
    değerleri içerir.

        global key, encrypted_aes_key, encrypted_hmac_key, ciphertext, mac, cipher_aes, original_message, n_value,
    e_value, salt

        # Kullanıcı arayüzünden gelen metnin alınması ve işlenmesi
        # Kullanıcı arayüzünden alınan metin, giriş kutusundan okunur, boşluk karakterleri temizlenir ve işlenir.
```

```
    message = mesaj_giris.get("1.0", "end-1c").strip()
    # Dosya adı
    dosya_adi = "sifreleme_degerleri.txt"

    if not message:  # Eğer mesaj kutusu boşsa

        # Şifreli ve deşifreli metin kutularının durumunun ayarlanması
        # Şifreli ve deşifreli metin kutularının durumu ayarlanır: aktif hale getirilir, temizlenir ve tekrar pasif
hale getirilir.

        desifreli_mesaj.config(state=NORMAL)
        desifreli_mesaj.delete("1.0", END)
        desifreli_mesaj.config(state=DISABLED)

        # Hata durumunun kontrol edilmesi
        # Eğer mesaj kutusu boşsa, kullanıcıya uygun bir hata mesajı gösterilir ve işlem sonlandırılır.
        auth_durum.config(text="Hata: Şifrelenecek metin kutusu boş.")
        return

    # Metnin tekrar alınması ve orijinal mesaj olarak saklanması
    # Mesaj tekrar alınır ve orijinal mesaj değişkenine atanır.
    message = mesaj_giris.get("1.0", "end-1c")
    original_message = message

    e_value = random.randint(
        10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000,
        10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    ) # Olası e değerlerinden birini seç

    key = RSA.generate(2048,e=e_value)    # Seçilen e değeriyle anahtar çiftini oluştur
    p_value=key.p
    q_value=key.q
    public_key = key.publickey()  # Genel anahtarı al
    n_value = key.n  # RSA n değeri
    e_value = key.e  # RSA e değeri
    salt = get_random_bytes(32)  # Rastgele bir tuz üret
    aes_key = PBKDF2(message.encode('utf-8'), salt, 32, count=90000, hmac_hash_module=SHA256)  # AES anahtarını türet
    hmac_key = PBKDF2(message.encode('utf-8'), salt, 32, count=9000, hmac_hash_module=SHA256)  # HMAC anahtarını
    # türet
    cipher_rsa = PKCS1_OAEP.new(public_key)  # RSA şifreleme nesnesi oluştur
    encrypted_aes_key = cipher_rsa.encrypt(aes_key)  # AES anahtarını RSA ile şifrele
    encrypted_hmac_key = cipher_rsa.encrypt(hmac_key)  # HMAC anahtarını RSA ile şifrele
    cipher_aes = AES.new(aes_key, AES.MODE_CBC)  # AES şifreleme nesnesi oluştur
    ciphertext = cipher_aes.encrypt(pad(message.encode('utf-8'), AES.block_size))  # Mesajı AES ile şifrele
    hmac = HMAC.new(hmac_key, digestmod=SHA256)  # HMAC nesnesi oluştur
    hmac.update(ciphertext)  # HMAC'ı şifreli metine uygula
    mac = hmac.digest()  # HMAC'ı tamamla

    # Şifreli metnin görüntülenmesi
    # Şifrelenmiş metin kutusu aktif hale getirilir, içeriği temizlenir, şifrelenmiş metin eklenir ve tekrar pasif
hale getirilir.
    sifreli_mesaj.config(state=NORMAL)
    sifreli_mesaj.delete("1.0", END)
    sifreli_mesaj.insert("1.0", ciphertext)
    sifreli_mesaj.config(state=DISABLED)

    hex_ciphertext = binascii.hexlify(ciphertext).decode('utf-8')  # Şifreli metni onaltılığa dönüştür

    # Şifrelenmiş metnin hexadecimal biçimde görüntülenmesi
    # Şifrelenmiş metin hexadecimal biçime dönüştürülür ve şifrelenmiş metin kutusuna eklenir.
```

```
sifreli_mesaj.config(state=NORMAL)
sifreli_mesaj.delete("1.0",bekle_ve_tamamlandi_yaz(sifreli_mesaj))
sifreli_mesaj.delete("1.0", END)
sifreli_mesaj.insert("1.0", hex_ciphertext)
sifreli_mesaj.config(state=DISABLED)


# Kullanıcıya anahtarlar ve diğer değerler hakkında bilgi verilmesi
# Anahtarlar ve diğer değerler, kullanıcıya uygun biçimde uyarı mesaj kutusuna eklenir.
uyari_mesaji.config(state=NORMAL)
uyari_mesaji.delete("1.0", END)
uyari_mesaji.insert("1.0",f"AES Anahtarı:\n\n{aes_key}\n\n"
                          f"HMAC Anahtarı:\n\n{hmac_key}\n\n"
                          f"Tuz Değeri:\n\n{salt}\n\n"
                          f"MAC:\n{mac}\n\n"
                          f"RSA n Değeri:\n\n{n_value}\n\n\n"
                          f"****************************************"
                          f"\nRSA e Değeri:\n\n{e_value}\n\n"
                          f"RSA p değeri:\n\n{p_value}\n\n"
                          f"RSA q değeri:\n\n{q_value}\n\n"
                          f"Doğrulama değeri:\n\n{p_value*q_value}\n\n"
                          f"Şifreli Metin(byte array):\n\n{ciphertext}")
uyari_mesaji.config(state=DISABLED)

# Dosya yolunu belirtmeden dosyayı aç ve değerleri dosyaya yaz
with open(dosya_adi, "w") as dosya:
    dosya.write(
        f"AES Anahtarı:\n\n{aes_key}\n\n"
        f"HMAC Anahtarı:\n\n{hmac_key}\n\n"
        f"Tuz Değeri:\n\n{salt}\n\n"
        f"MAC:\n{mac}\n\n"
        f"RSA n Değeri:\n\n{n_value}\n\n\n"
        f"****************************************"
        f"\nRSA e Değeri:\n\n{e_value}\n\n"
        f"RSA p değeri:\n\n{p_value}\n\n"
        f"RSA q değeri:\n\n{q_value}\n\n"
        f"Doğrulama değeri:\n\n{p_value * q_value}\n\n"
        f"Şifreli Metin(byte array):\n\n{ciphertext}"
    )
# Deşifreleme fonksiyonu
def desifrele():
    global key, cipher_aes, original_message

    # Deşifreleme işlemi için gerekli bilgilerin alınması
    # Kullanıcıdan şifreli metin alınır ve işlenir.
    user_input = kullanici_sifreli_giris.get("1.0", "end-1c").strip()

    if not user_input:  # Eğer kullanıcı şifreli metin kutusu boşsa

        # Deşifrelenmiş metin kutusunun durumunun ayarlanması
        # Deşifrelenmiş metin kutusu aktif hale getirilir, içeriği temizlenir ve tekrar pasif hale getirilir.
        sifreli_mesaj.config(state=NORMAL)
        sifreli_mesaj.delete("1.0", END)
        sifreli_mesaj.config(state=DISABLED)

        # Hata durumunun kontrol edilmesi
        # Eğer kullanıcı şifreli metin kutusunu boş bırakmışsa, uygun bir hata mesajı gösterilir ve işlem
sonlandırılır.
        auth_durum.config(text="Hata: Şifreli metin kutusu boş.")
        return
    try:
```

```
        user_ciphertext = binascii.unhexlify(user_input)  # Onaltılık girişi ikili forma dönüştür

    # Girilen metnin hexadecimal biçimde kontrol edilmesi
    # Eğer girilen metin hexadecimal biçimde değilse, uygun bir hata mesajı gösterilir ve işlem sonlandırılır.
    except (ValueError, binascii.Error):
        auth_durum.config(text="Hata: Girilen metin onaltılık formatta değil.")
        return

    # Şifreli metinlerin karşılaştırılması
    # Kullanıcı tarafından girilen şifreli metin, orijinal şifrelenmiş metinle karşılaştırılır.

    if user_ciphertext == ciphertext:
            try:
                cipher_rsa = PKCS1_OAEP.new(key)  # RSA şifreleme nesnesi oluştur
                decrypted_aes_key = cipher_rsa.decrypt(encrypted_aes_key)  # AES anahtarını deşifre et
                cipher_aes = AES.new(decrypted_aes_key, AES.MODE_CBC, iv=cipher_aes.iv)  # AES şifreleme nesnesi
oluştur
                decrypted_text = unpad(cipher_aes.decrypt(user_ciphertext), AES.block_size).decode('utf-8')  #
Şifreli metni çöz

                # Deşifreleme işleminin gerçekleştirilmesi
                # Kullanıcının şifreli metnini deşifrelemek için gerekli işlemler gerçekleştirilir.
                desifreli_mesaj.config(state=NORMAL)
                desifreli_mesaj.delete("1.0", bekle_ve_tamamlandi_yaz(desifreli_mesaj))

                desifreli_mesaj.delete("1.0", END)  # 1.0 : satır sütun konumu
                desifreli_mesaj.insert("1.0", decrypted_text)  # Çözülmüş metni göster
                desifreli_mesaj.config(state=DISABLED)

            except (ValueError, TypeError, binascii.Error) as e:
                auth_durum.config(text=f"Deşifreleme sırasında hata oluştu: {str(e)}")
                auth_durum.delete("1.0", END)
                return

            karsilastirma_durumu.config(text="Kullanıcı şifresi orijinal şifre ile eşleşiyor.")
    else:
            karsilastirma_durumu.config(text="Kullanıcı şifresi orijinal şifre ile eşleşmiyor.")
            return


# Kullanıcı arayüzü oluşturulması
# Tkinter penceresini oluştur
root = Tk()

# Pencere boyutunu ayarla
root.geometry("1400x1004+200+30")

# Pencere arka plan rengini ayarla
root.configure(background="#57cac3")

# Yazı tipi stillerini tanımla
font_style = ("Helvetica", 14)
font_style2 = ("Helvetica", 15)
heigh=7
# Pencere başlığını belirle
root.title("Şifreleme ve Deşifreleme Uygulaması")

# Ana çerçeve oluştur
ana_cerceve = Frame(root)
ana_cerceve.pack(side=LEFT)  # Ana çerçeve ana pencerenin sol tarafında yer alacak

# Şifrelenecek metin için etiket ve giriş alanı oluştur
```

```
Label(ana_cerceve, text="Şifrelenecek Metin:", font=font_style).pack()
mesaj_giris = Text(ana_cerceve, height=heigh, font=font_style2)
mesaj_giris.pack()
Button(ana_cerceve, text="Şifrele", command=sifrele, font=font_style).pack()

# Şifreli metin için etiket ve alan oluştur
Label(ana_cerceve, text="Şifreli Metin:", font=font_style).pack()
sifreli_mesaj = Text(ana_cerceve, height=heigh, state=DISABLED, font=font_style2)
sifreli_mesaj.pack()

# Kullanıcıdan alınacak şifreli metin için etiket ve giriş alanı oluştur
Label(ana_cerceve, text="Kullanıcının Gireceği Şifreli Metin (hex):", font=font_style).pack()
kullanici_sifreli_giris = Text(ana_cerceve, height=heigh, font=font_style2)
kullanici_sifreli_giris.pack()
Button(ana_cerceve, text="Deşifrele", command=desifrele, font=font_style).pack()

# Deşifrelenmiş metin için etiket ve alan oluştur
Label(ana_cerceve, text="Deşifre Edilmiş Metin:", font=font_style).pack()
desifreli_mesaj = Text(ana_cerceve, height=heigh, state=DISABLED, font=font_style2)
desifreli_mesaj.pack()

# Doğrulama durumu için etiket oluştur
auth_durum = Label(ana_cerceve, text="---------", font=font_style)
auth_durum.pack()

# Karşılaştırma durumu için etiket oluştur
karsilastirma_durumu = Label(ana_cerceve, text="---------", font=font_style)
karsilastirma_durumu.pack()

# Açıklama çerçevesi oluştur
aciklama_cerceve = Frame(root)
aciklama_cerceve.pack(side=RIGHT)  # Açıklama çerçevesi ana pencerenin sağ tarafında yer alacak

# Açıklama etiketi oluştur ve yerleştir
Label(aciklama_cerceve, text="Açıklama: ", font=font_style).grid(row=0, column=0, sticky=W)  # Grid yöneticisi
# kullanılarak etiket sağ üst köşeye yerleştirilecek

# Açıklama text alanı oluştur ve yerleştir
uyari_mesaji = Text(aciklama_cerceve, height=38, width=85, wrap=WORD, font=font_style2)
uyari_mesaji.config(state=DISABLED)
uyari_mesaji.grid(row=1, column=0, sticky=W)  # Grid yöneticisi kullanılarak açıklama texti etiketin altına
yerleştirilecek


# Uygulamayı çalıştır
root.mainloop()
```