S-DES 加解密程序开发手册

1. 简介

本开发手册提供了一个基于 S-AES（Simplified Advanced Encryption Standard）算法的加解密程序的详细接口文档，用于对数据进行简单小规模加密和解密操作。通过 S-AES 算法对信息加密，实现保证信息安全，保障信息传输过程中的可靠性的功能。同时确保其在传输过程中不被未经授权的人员所访问。

在次开发手册中，将包含以下内容：

（1） 组件及接口概述：介绍了加密解密程序的各个组件和对外提供的接口。

（2） 使用示例：提供了一些示例代码，展示了如何使用加密解密程序完成加密和解密操作。

（3） 注意事项：列出了在使用加密解密程序时需要注意的事项和建议。

通过阅读此开发手册，您将了解如何正确使用 S-DES 加密解密程序，以及如何保护您的数据安全。请确保遵守安全性原则，并妥善保管生成的密钥，一面数据被未授权的人员获取。

2. 组件及接口概述

本程序包含以下组件及其对应的接口：

2.1 spl(input)

该函数用于将输入的文本字符串转换为二进制数据，以便进行加密和解密操作。它将文本中的每个字符转换为 ASCII 码，然后将 ASCII 码转化为二进制表示。

2.2 binary_array_to_int(binary_array)

该函数用于将二进制数组转化为整数值

2.3 使用 AES 加密和解密：

代码提供了一些示例用法，包括常规加解密和 ASCII 加解密。这些示例演示了如何使用 AES 加密算法对数据进行加密和解密。

2.4 多重加密：

提供了双重加密和三重加密的示例。这些示例演示了如何多次使用不同的 AES 密钥对数据进行加密和解密。

2.5 CBC 模式：

代码演示了密码分组链接模式（CBC 模式）的使用，其中明文数据被分成块并使用 AES 加密算法进行加密。CBC 模式在加密前会对每个块进行异或操作，以增加安全性。

2.6 GUI 界面展示：

调用 python 中的 tkinter 库，设计并实现程序的 UI 界面。

3. 使用示例（基于 python）

以编码为 Ascii 模式下的加解密算法进行示例：

```
import binascii
import numpy as np
import aes128
import time
```

```
def spl(input):
    transform = []
    for i in range(len(input)):
        transform.append(ord(input[i]))
    data = []
    for i in range(len(transform)):
        b = bin(transform[i]).replace('0b', '')
        b = b.zfill(8)
        for j in range(len(b)):
            data.append(int(b[j]))
    data1 = np.array(data)
    data1.resize((len(input), 8))
    return data1
```

```
def binary_array_to_int(binary_array):
```

```
    if len(binary_array) != 4:
        raise ValueError("二进制数组长度必须为 4 位")
    decimal_value = 0
    for bit in binary_array:
        decimal_value = (decimal_value << 1) | bit
```

```
    return decimal_value
```

```
key = [0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
```

```
#ascii 加解密
pla_str = "abcd"
pla = spl(pla_str)
print(pla)
temp = []
crypted_data = []
for byte in pla:
    temp.append(binary_array_to_int(byte[:4]))
    temp.append(binary_array_to_int(byte[4:]))
    if len(temp) == 4:
        crypted_part = aes128.encrypt(temp, key)
        crypted_data.extend(crypted_part)
        del temp[:]
else:
    # padding v1
    # crypted_data.extend(temp)
    # padding v2
    if 0 < len(temp) < 4:
        empty_spaces = 4 - len(temp)
        for i in range(empty_spaces - 1):
            temp.append(0)
        temp.append(1)
        crypted_part = aes128.decrypt(temp, key)
        crypted_data.extend(crypted_part)

print(crypted_data)
```

```
decrypted_data = []
temp = []
for byte in crypted_data:
    temp.append(byte)
    if len(temp) == 4:
        decrypted_part = aes128.decrypt(temp, key)
```

```
        decrypted_data.extend(decrypted_part)
        del temp[:]
else:
    if 0 < len(temp) < 4:
        empty_spaces = 4 - len(temp)
        for i in range(empty_spaces - 1):
            temp.append(0)
        temp.append(1)
        decrypted_part = aes128.encrypt(temp, key)
        decrypted_data.extend(decrypted_part)
decrypted_str = ""
for i in range(len(decrypted_data)):
    if i % 2 == 0:
        t = 16 * decrypted_data[i] + decrypted_data[i + 1]
        decrypted_str += chr(t)
```

```
print(decrypted_str)
```

4. 注意事项

在 GUI 界面中输入密钥的时候直接输入密钥数字，不需要按特定格式输入。但是需要保证输入的密钥位数为十。

请每一次通讯时妥善保管密钥，不要将密钥泄露给他人。

本程序仅适用于小规模的数据加密需求，不建议用于对大量或敏感数据进行加密。