

通关情况测试

3.1 第1关：基本测试

在"加解密"标签页中,可以进行以下操作:

- 输入16进制的明文和密钥,点击"加密"按钮进行加密,点击"解密"按钮进行解

加密

解密

S-AES 加密/解密

加解密

说明

● CBC模式 S-AES加密/解密

十六进制

ASCII

双重加密

三重加密

明文

密钥

加密

解密

结果 (十六进制或字符)

0xb601

说明:

- 我们支持十六进制 (即基本的S-AES加密), 双重加密, 三重加密仅支持16-bits的16进制的输入 (例如0f0f).
- 密钥也是16-bits的16进制 (例如0f0f), 对于双重加密, 三重加密的密钥分别需要32-bits (例如16进制000214a), 48-bits (例如16进制1112223333), 我们分别前4个作为key1, 再4个作为key2...
- ASCII加密支持任意长度的输入, 密钥也是16-bits的16进制 (例如0f0f).

S-AES 加密/解密

加解密

说明

● CBC模式 S-AES加密/解密

十六进制

ASCII

双重加密

三重加密

明文

密钥

加密

解密

结果 (十六进制或字符)

0xf00

说明:

- 我们支持十六进制 (即基本的S-AES加密), 双重加密, 三重加密仅支持16-bits的16进制的输入 (例如0f0f).
- 密钥也是16-bits的16进制 (例如0f0f), 对于双重加密, 三重加密的密钥分别需要32-bits (例如16进制000214a), 48-bits (例如16进制1112223333), 我们分别前4个作为key1, 再4个作为key2...
- ASCII加密支持任意长度的输入, 密钥也是16-bits的16进制 (例如0f0f).

3.2 第2关：交叉测试

基本加密:

我们组

其他组

S-AES 加密/解密

加解密

说明

● CBC模式 S-AES加密/解密

十六进制

ASCII

双重加密

三重加密

明文

密钥

加密

解密

结果 (十六进制或字符)

0xb601

说明:

- 我们支持十六进制 (即基本的S-AES加密), 双重加密, 三重加密仅支持16-bits的16进制的输入 (例如0f0f).
- 密钥也是16-bits的16进制 (例如0f0f), 对于双重加密, 三重加密的密钥分别需要32-bits (例如16进制000214a), 48-bits (例如16进制1112223333), 我们分别前4个作为key1, 再4个作为key2...
- ASCII加密支持任意长度的输入, 密钥也是16-bits的16进制 (例如0f0f).

Encryption and Decryption

Select form:

Binary

Message:

000011100000000

Key:

0010110101010101

Encrypt

Decrypt

CipherText: 1011011000000001

密钥: 2d55

明文: 0f00

密文: b601

密钥:

0010110101010101

明文:

0000111100000000

密文:

1011011000000001

双重加密:

No. 1 / 6

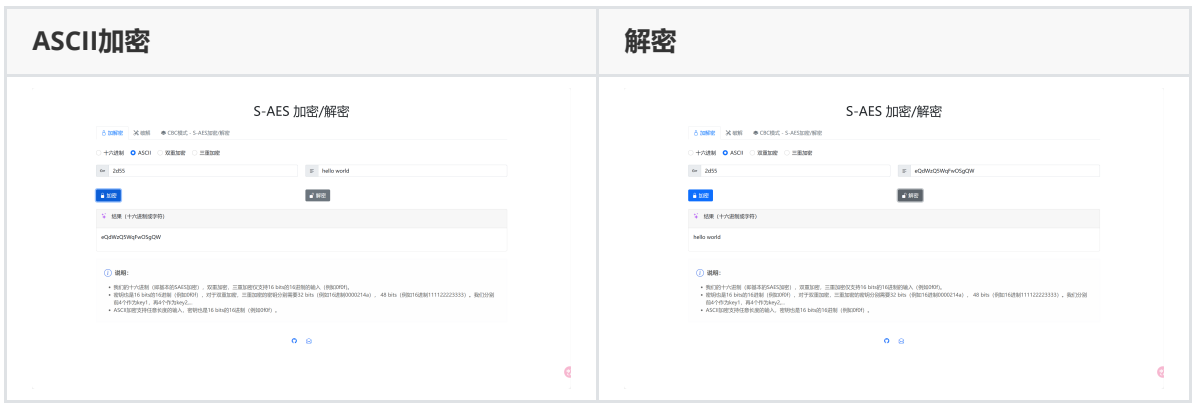
我们组	其他组
<div><p>S-AES 加密/解密</p><p>十六进制 ASCII 三重加密</p><p>2d55abcd 1234</p><p>5fd5</p><p>结果 (十六进制密文): 5fd5</p><p>说明:</p><ul style="list-style-type: none">基本的十六进制 (即基本的S-AES加密) , 三重加密, 三重加密仅支持16 bit(2字节)的明文 (即00000000)。密文由16 bit(2字节) (即00000000) , 对于三重加密, 三重加密的密文仅支持16 bit (即0000000000000000) , 48 bit (即000000001111111111111111) , 密文由16 bit(2字节) (即00000000) , 密文由16 bit(2字节) (即00000000)。ASCII加密支持任意长度的明文, 密文由16 bit(2字节) (即00000000)。</div>	<div><p>Multiple Encryption and Decryption</p><p>Select form: Double en-decryption</p><p>Message: 0001001000110100</p><p>Key: 001011010101010101010101111001101</p><p>Double Encrypt Double Decrypt</p><p>CipherText: 0101111111010101</p></div>
<div><p>密钥: 2d55abcd</p><p>明文: 1234</p><p>密文: 5fd5</p></div>	<div><p>密钥:</p><p>001011010101010101010101111001101</p><p>明文: 0001001000110100</p><p>密文: 0101111111010101</p></div>

三重加密:

我们组	其他组
<div><p>S-AES 加密/解密</p><p>十六进制 ASCII 三重加密</p><p>B39ADECD8A27 D72C</p><p>1911</p><p>结果 (十六进制密文): 1911</p><p>说明:</p><ul style="list-style-type: none">基本的十六进制 (即基本的S-AES加密) , 三重加密, 三重加密仅支持16 bit(2字节)的明文 (即00000000)。密文由16 bit(2字节) (即00000000) , 对于三重加密, 三重加密的密文仅支持16 bit (即0000000000000000) , 48 bit (即000000001111111111111111) , 密文由16 bit(2字节) (即00000000) , 密文由16 bit(2字节) (即00000000)。ASCII加密支持任意长度的明文, 密文由16 bit(2字节) (即00000000)。</div>	<div><pre>1 triple_key= '101100111001101010111011001101101101100100111' 2 print(f'本次SAES加密密钥长度为: {len(triple_key)}') 3 triple_saes = SAES(key=triple_key) 4 triple_plaintext = '1101011100101100' 5 print(f'本次SAES加密明文为: {triple_plaintext}') 6 triple_encrypted_ciphertext = triple_saes.encrypt(triple_plaintext) 7 print(f'通过SAES加密后的密文为: {triple_encrypted_ciphertext}') 8 triple_decrypted_plaintext = triple_saes.decrypt(triple_encrypted_ciphertext) 9 print(f'通过SAES解密后的明文为: {triple_decrypted_plaintext}')</pre><p>Executed at 2023.10.23 21:30:08 in 5ms</p><p>本次SAES加密密钥长度为: 48</p><p>本次SAES加密明文为: 1101011100101100</p><p>通过SAES加密后的密文为: 0001100100010001</p><p>通过SAES解密后的明文为: 1101011100101100</p></div>
<div><p>密钥: B39ADECD8A27</p><p>明文: D72C</p><p>密文: 1911</p></div>	<div><p>密钥:</p><p>101100111001101010111011001101101101100100111</p><p>明文: 1101011100101100</p><p>密文: 0001100100010001</p></div>

3.3 第3关：扩展功能

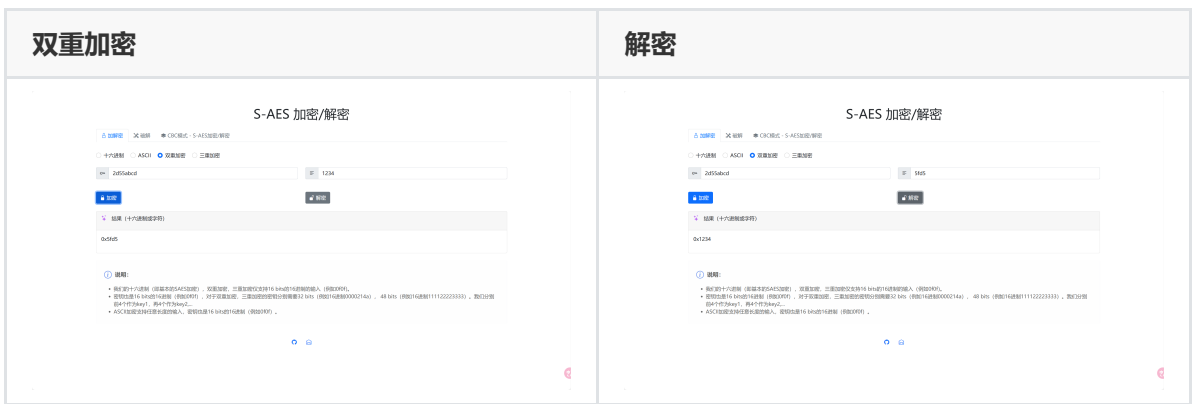
- 选择"ASCII"选项,可以输入任意长度的ASCII字符串进行加密和解密



3.4 第4关：多重加密

3.4.1 双重加密

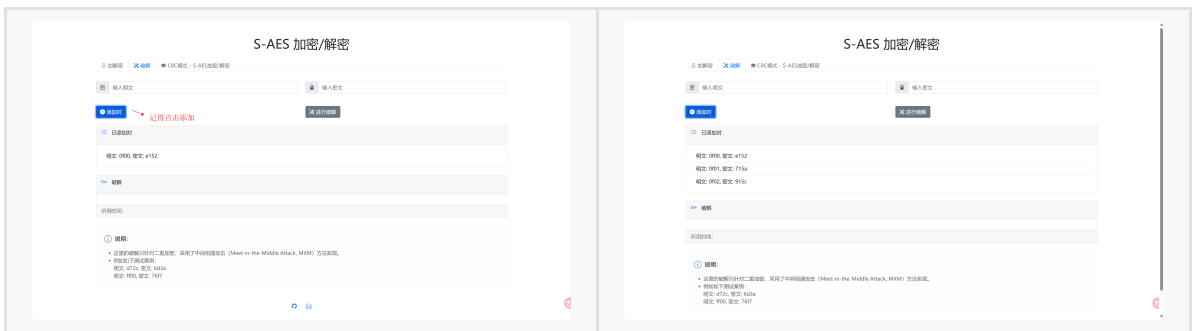
- 选择"双重加密",可以输入16进制明文和32位十六进制密钥,进行双重加密和解密



3.4.2 中间相遇攻击

在“破解”标签页中,可以进行中间相遇攻击,步骤如下:

- 输入明文和密文组成一对,点击"添加对"按钮
- 重复上述步骤,添加多个明文文对



- 点击"进行破解"按钮,会显示破解得到的密钥 k1 和 k2 ,以及破解耗时



3.4.3 三重加密

使用48bits(K1+K2+K3)的模式进行三重加解密

- 选择“三重加密”,可以输入16进制明文和48位十六进制密钥,进行三重加密和解密



3.5 第5关：工作模式

在“CBC模式”标签页中,可以进行CBC模式的加解密:

- 输入16进制密钥和初始向量 IV
- 在长明文输入框中输入要加密的明文
- 点击“CBC加密”按钮进行加密,结果会显示在结果区域



- 可以点击展开按钮,添加或者修改密文

- 点击“CBC解密”按钮进行解密,查看修改前后解密结果的比较



讨论：

Key Used for Encryption: 0x1234

Initialization Vector (IV): 0x5678

Tampering	Original Ciphertext	Tampered Ciphertext	Decrypted Tampered Ciphertext
None	0xf2cf10e25b8dfd0a	N/A	0x9abc9abcdef01234
4 bits	0xf2cf10e25b8dfd0a	0x02cf10e25b8dfd0a	0x6abe6abcdef01234
8 bits	0xf2cf10e25b8dfd0a	0x0dcf10e25b8dfd0a	0x675e65bcdef01234
12 bits	0xf2cf10e25b8dfd0a	0x0d3f10e25b8dfd0a	0x6d7e654cdef01234
16 bits	0xf2cf10e25b8dfd0a	0x0d3010e25b8dfd0a	0x4d716543def01234
32 bits	0xf2cf10e25b8dfd0a	0x0d30ef1d5b8dfd0a	0x4d719659210f1234

分析如下：

1. **错误传播特性：**从结果中，我们可以看到CBC模式的一个显著特点，即一个加密块的篡改会影响到解密的该块和下一个块。这称为错误传播特性。比如当我们篡改第一个块时，解密的第一个块和第二块都受到了影响。
2. **一致性的缺失：**尽管错误传播到后续块，但篡改加密的块并不能以有意义的方式影响解密的块。换句话说，一个小小的篡改（例如4比特）可能会导致完全不同的解密输出，而不是仅仅4比特的差异。这也是为什么篡改加密消息是危险的，因为它会导致完全不可预测的输出。
3. **影响的持续性：**当我们篡改32位时，尽管第一个块和第二块都受到了影响，但从第三块开始，解密的输出与原始明文相同。这证明了CBC模式中错误传播的“有限”特性，即一个错误只会影响其本身和下一个块。
4. **安全性与篡改检测：**CBC模式本身并不提供明文的完整性或篡改检测。从结果中，我们可以看到，即使对加密文本进行了小小的篡改，也可能不容易被接收者察觉，除非他们期望的解密输出与实际的解密输出存在显著差异。
5. **篡改与噪声：**篡改在真实世界中可能被视为噪声或数据损坏。CBC模式对于随机噪声具有某种“弹性”，因为噪声只影响两个块，而不是整个消息。

总之，CBC模式提供了一种方式，使篡改的加密消息在解密时产生不可预测的输出，并且篡改会传播到后续的块，但其影响是有限的。为了提高安全性，应该考虑使用其他机制来确保数据的完整性和篡改检测。

