

测试结果

1.实验概述

本次实验是据"信息安全导论"课程讲述的 S-AES 算法，以及所学知识基础上，使用 Python+QT 编程实现加密，解密算法，以及后续的测试闯关。

2.测试闯关

2.1 第 1 关：基本测试

根据 S-AES 算法编写和调试程序，提供 GUI 解密支持用户交互。输入可以是 16bit 的数据和 16bit 的密钥，输出是 16bit 的密文。

(1) 用户交互界面：

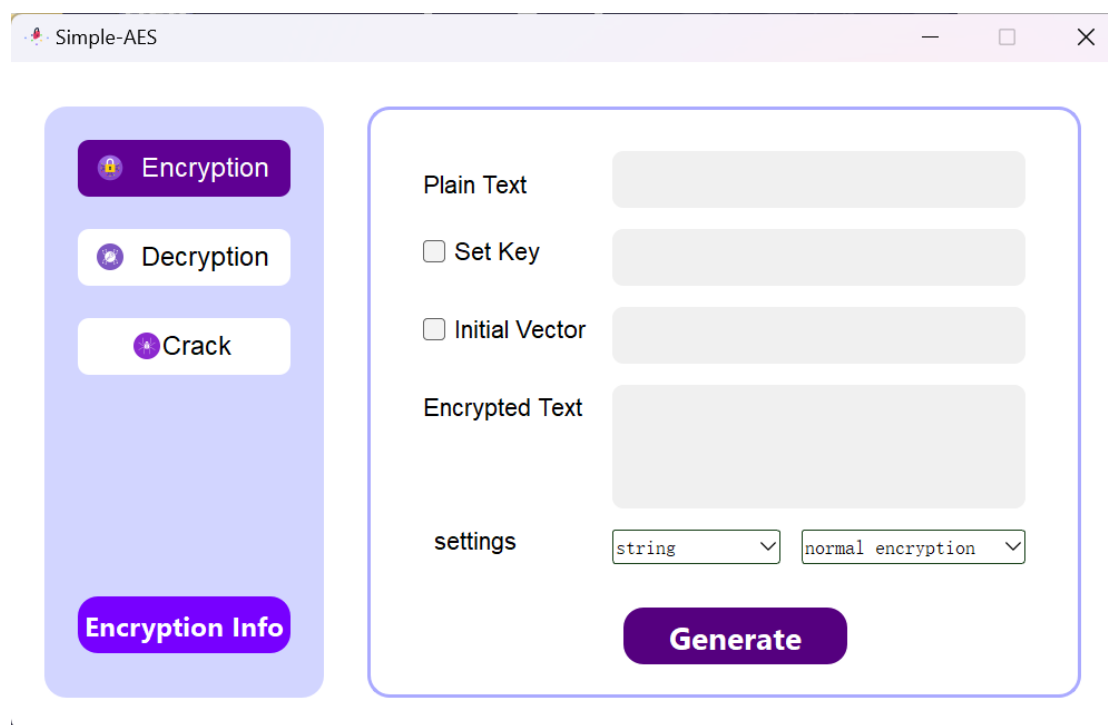


图 1 用户交互界面

(2) 对同一对明密文对进行加/解密操作：

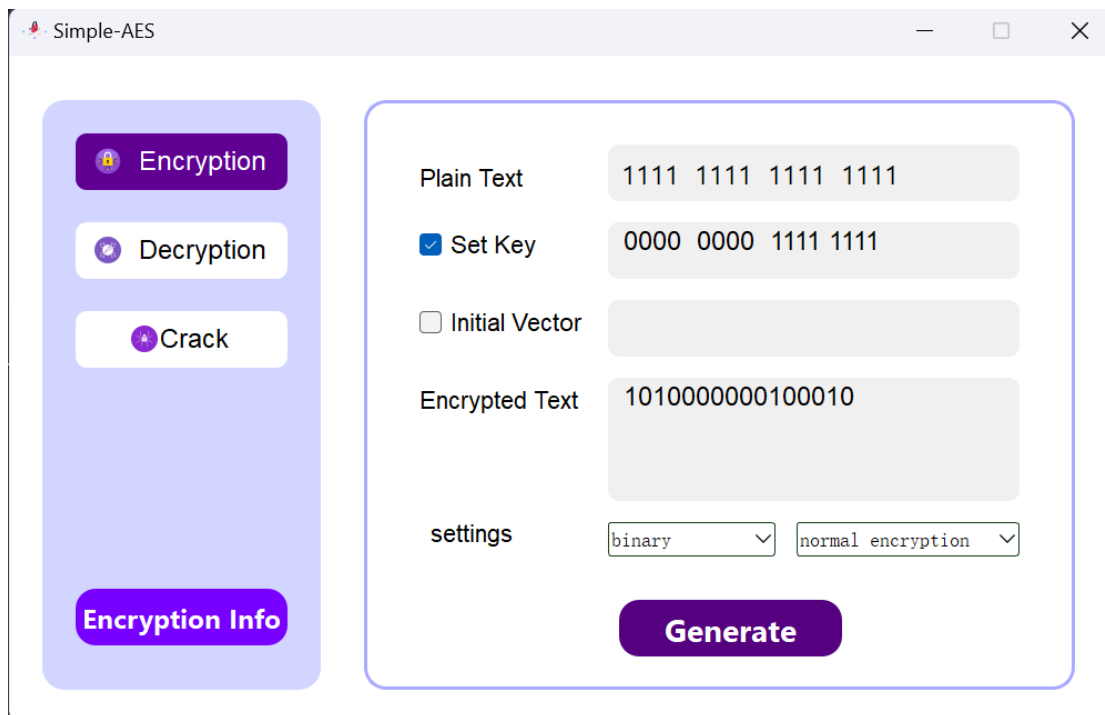


图 2 加密

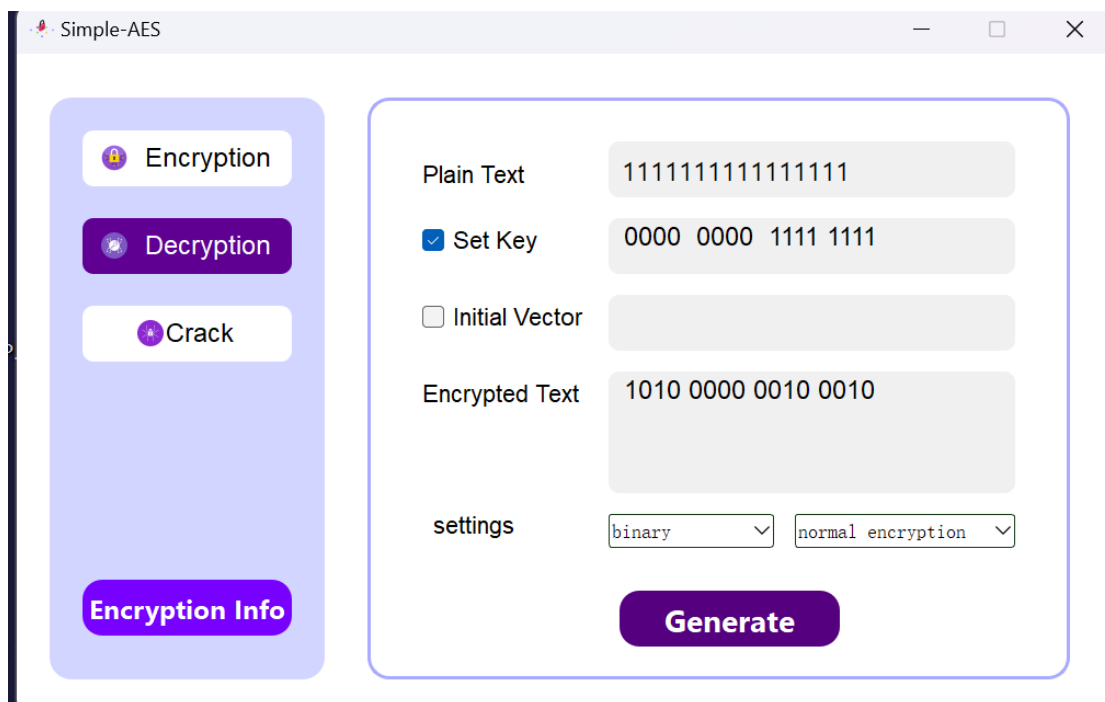


图 3 解密

测试结果：加解密完成，测试通过。

2.2 第 2 关：交叉测试

考虑到是算法标准，所有人在编写程序的时候需要使用相同算法

流程和转换单元(P-Box、S-Box 等)，以保证算法和程序在异构的系统或平台上都可以正常运行。我们小组与 epsilon 小组进行交叉测试：

我们都使用明文：0000 0111 0011 1000 ，密钥：1010 0111 0011 1011 ，进行密文生成的测试，如果结果相同则测试通过：
我们小组测试结果：

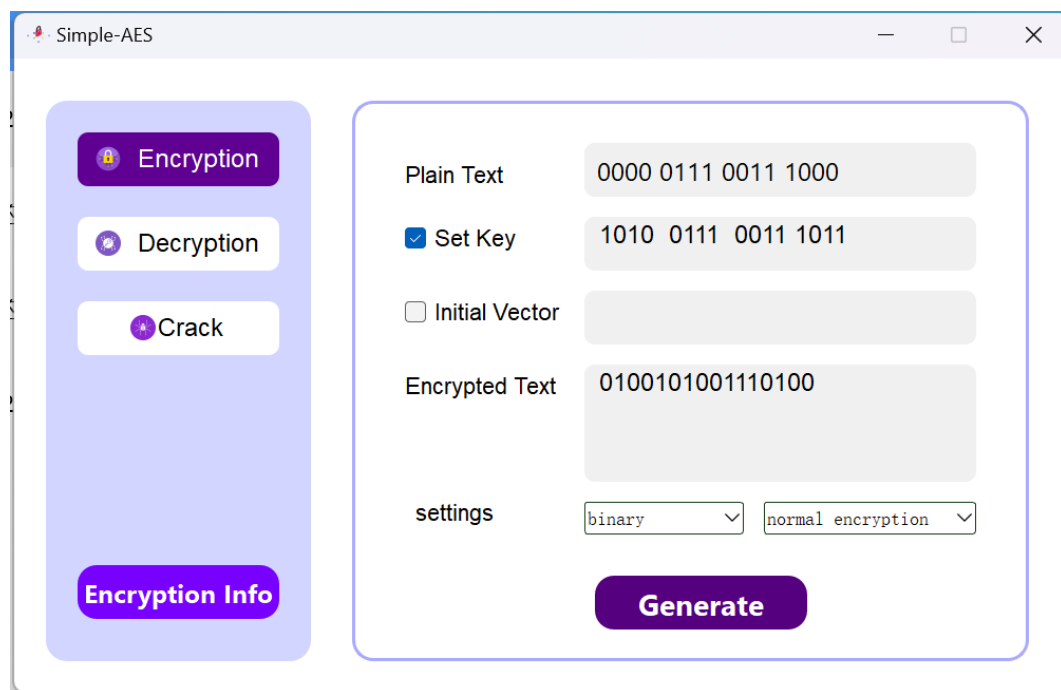


图 3 我们小组测试结果

epsilon 小组测试结果：

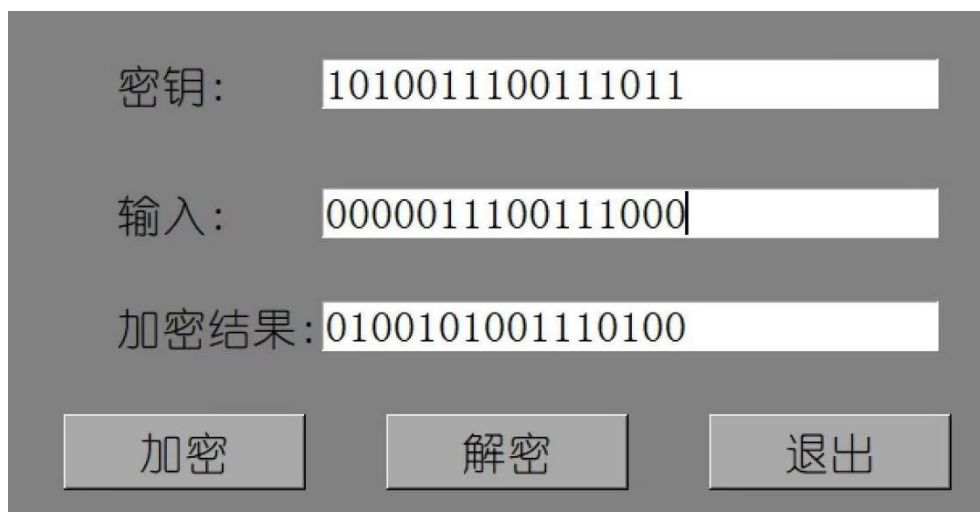


图 4 epsilon 小组测试结果

测试结果：双方的加密结果一致，测试通过。

2.3 第 3 关：扩展功能

考虑到向实用性扩展，加密算法的数据输入可以是 ASCII 编码字符串，对应地输出也可以是 ASCII 字符串。在我们的界面中设计了以符文输出的选项，在这里只需要调整选择就能输出相应的符文加密结果：

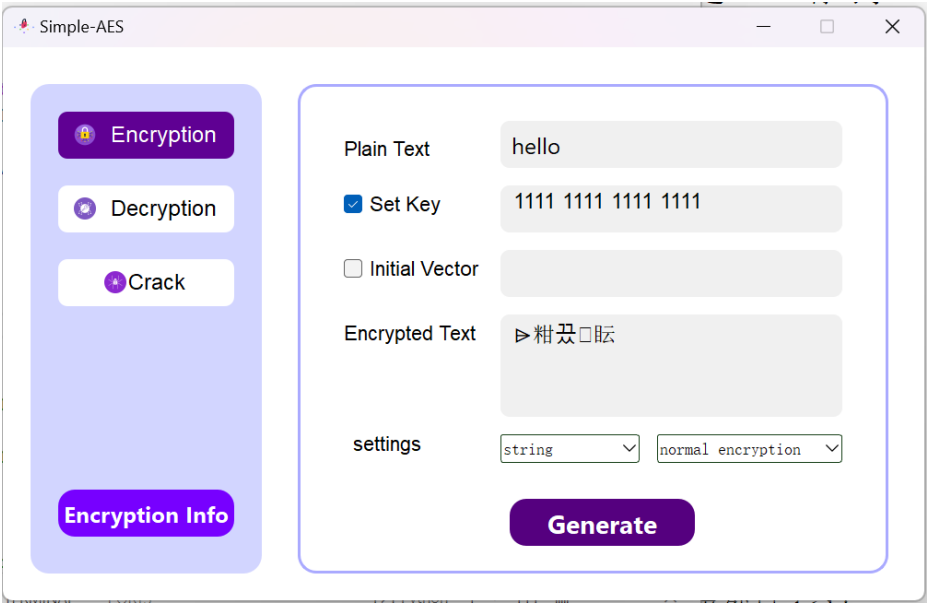


图 5 字符类型加密结果

从测试结果可以看出测试通过。

2.4 第 4 关：多重加密

2.4.1 双重加密

将 S-AES 算法通过双重加密进行扩展，分组长度仍然是 16 bits，但密钥长度为 32 bits，以下为测试结果：

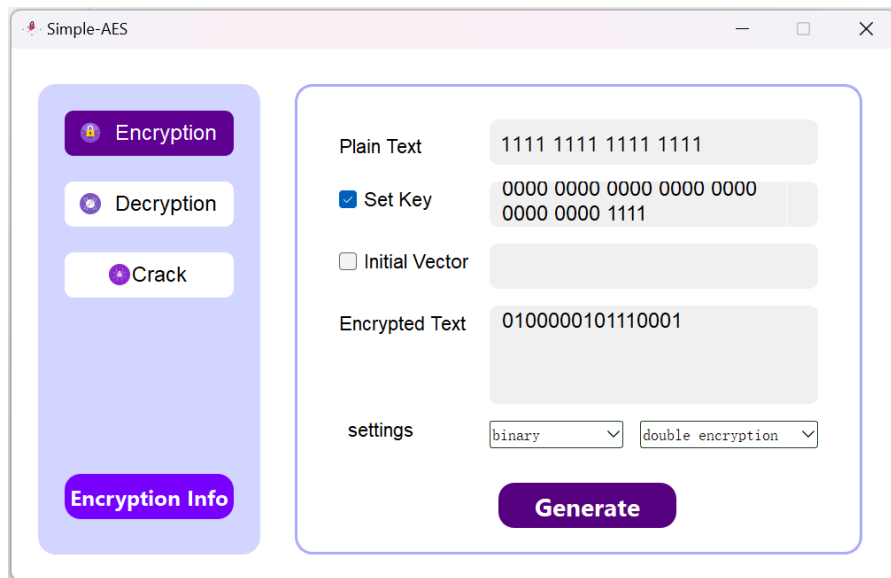


图 6 双重加密 加密

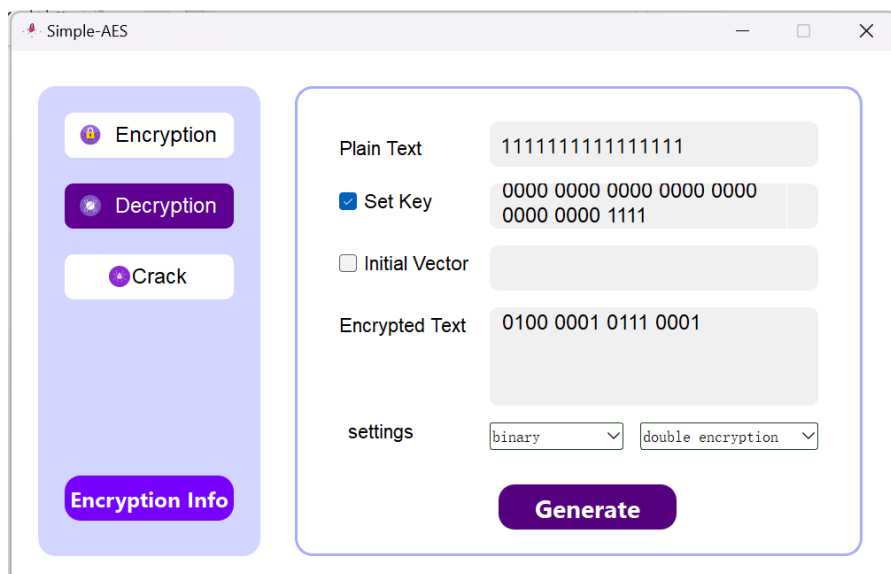


图 7 双重加密 解密

从图中可以看出测试通过，同时与其他小组进行了交叉测试，加密解密结果一致。

2.4.2 中间相遇攻击

假设找到了使用相同密钥的明、密文对(一个或多个)，请尝试使用中间相遇攻击的方法找到正确的密钥 $Key(K1+K2)$ ，这是一种针对双重加密算法的攻击，需要已知的明文和密文对，在本质上明文被加密在双重加密中产生一个中间值，密文被解密在双重加密中产生一个

中间值，找到中间值则能测出结果。以下为测试结果：

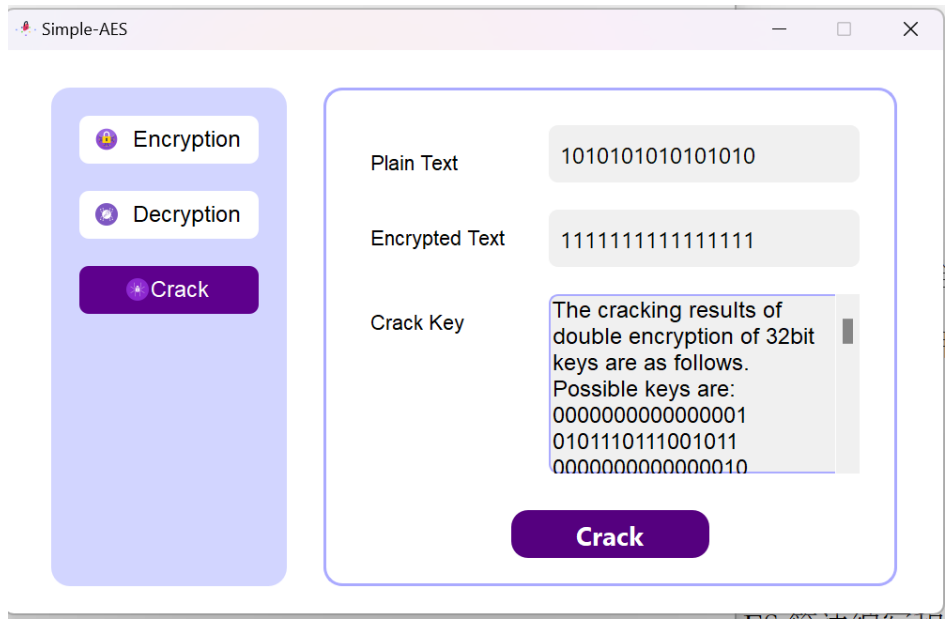


图 8 中间相遇攻击

在中间相遇攻击中找到较多的密钥对，每次得到结果为 1 组 2 个 16bit 的密钥 k_1 和 k_2 ，在这个过程中也可以输入多对明密文对，得到共同密钥。在解密的过程中我们可以看出来解密的时间明显增长，说明时间代价较高。

这次与另外一个小组进行了三对明密文对的交叉测试，测试结果一致，测试通过。

2.4.3 三重加密

将 S-AES 算法通过三重加密进行扩展，我们使用的 48bits($K_1+K_2+K_3$)的模式进行三重加解密：

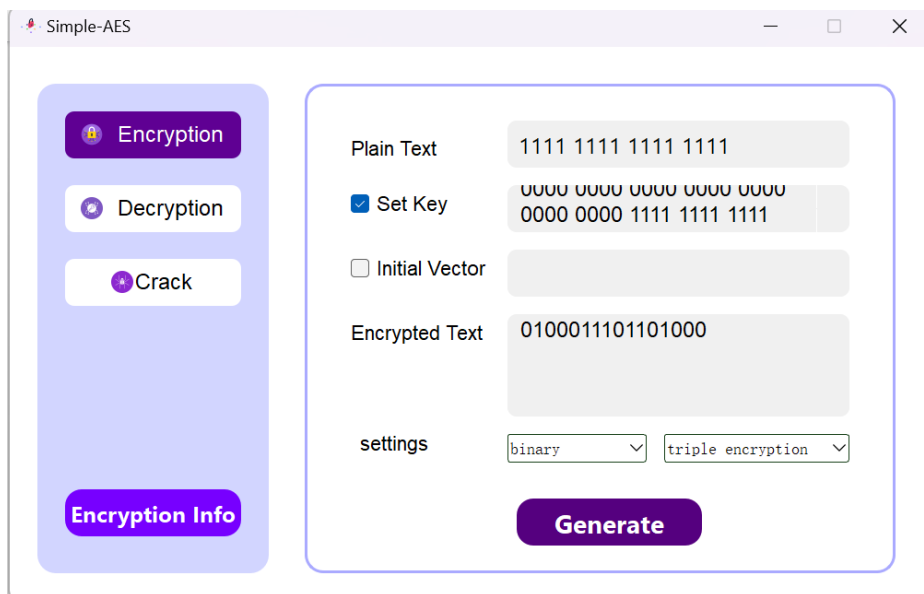


图 9 三重加密 加密

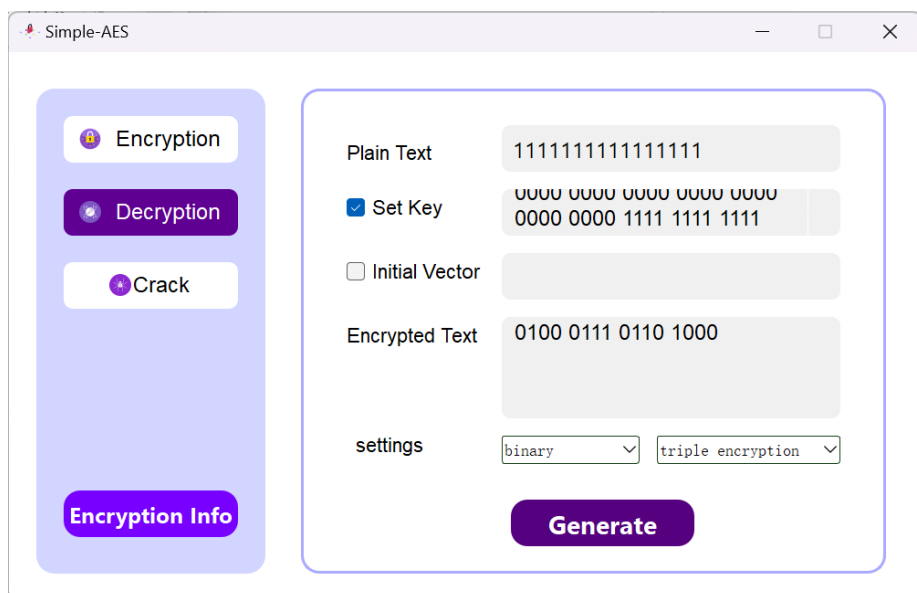


图 10 三重加密 解密

从测试的情况可以看出测试成功，同时与其他小组进行了交叉测试，测试结果显示结果一致，测试通过。

2.5 第 5 关：工作模式

基于 S-AES 算法，使用密码分组链(CBC)模式对较长的明文消息进行加密。注意初始向量(16 bits) 的生成，并需要加解密双方共享。在 CBC 模式下进行加密，并尝试对密文分组进行替换或修改，

然后进行解密，请对比篡改密文前后的解密结果。

这是再 CBC 模式下的加密情况，明文为三个 16bit 明文 1111 0000 1111 0000 1111 1111 1111 0000 0000 0000 0000 ， 密钥为 0000 0000 1111 1111 ， 最终密文结果为： 0110000010000001 0101001110000101 0101001111011101

The image shows a web application titled "Simple-AES". On the left is a sidebar with three buttons: "Encryption" (selected), "Decryption", and "Crack". Below these is an "Encryption Info" button. The main area contains the following fields and controls:

- Plain Text:** 1111 1111 0000 0000 0000 0000
- Set Key:** ☒ 0000 0000 1111 1111
- Initial Vector:** ☐
- Encrypted Text:** 0110000010000001
0101001110000101
0101001111011101
- settings:** "binary" (dropdown) and "normal encryption" (dropdown)
- Generate:** A large purple button at the bottom.

图 11 CBC 模式加密

然后对加密结果进行篡改，修改其中的一个比特位，修改第一个比特位 1110000010000001 0101001110000101 0101001111011101，发现结果出现了很大的偏差，明文组变为： 0101010110100011 0111111111111111 0000000000000000 ， 只篡改了第一组密文结果，全部的明文都出现了错误！

The image shows a web application titled "Simple-AES". On the left is a sidebar with three buttons: "Encryption" (highlighted), "Decryption", and "Crack". Below these is an "Encryption Info" button. The main area contains the following fields and controls:

- Plain Text:** 1111111111 0000000000000000
- Set Key:** ☒ (checked), value: 0000 0000 1111 1111
- Initial Vector:** ☐ (unchecked), empty field
- Encrypted Text:** 1110000010000001
01010011110000101
0101001111011101|
- settings:** "binary" (dropdown), "normal encryption" (dropdown)
- Generate:** A large purple button at the bottom right.

图 12 篡改内容得到的结果

测试结果：得到的明文出现很大区别，说明 CBC 模式对于一组密文一个小改变都会导致所有明文组出现变化，对密文进行篡改会导致解密结果与明文相差较大，完成工作模式的测试。